Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys

WEI DONG, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore
JUANRU FANG, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong
KE YI, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong
YUCHAO TAO, Duke University, Durham, United States
ASHWIN MACHANAVAJJHALA, Computer Science, Duke University, Durham, United States

Answering SPJA queries under differential privacy (DP), including graph pattern counting under node-DP as an important special case, has received considerable attention in recent years. The dual challenge of foreign-key constraints combined with self-joins is particularly tricky to deal with, and no existing DP mechanisms can correctly handle both. For the special case of graph pattern counting under node-DP, the existing mechanisms are correct (i.e., satisfy DP), but they do not offer nontrivial utility guarantees or are very complicated and costly. In this article, we propose two mechanisms for solving this problem with both efficiency and strong utility guarantees. The first mechanism, called *R2T*, is simple and efficient, while achieving *down-neighborhood optimality* with a logarithmic optimality ratio. Down-neighborhood optimality is a new notion of optimality that we introduce for measuring the utilities of DP mechanisms, which can be considered as a natural relaxation of instance optimality, and it is especially suitable for functions with a large or unbounded sensitivity. Our second mechanism further reduces the optimality ratio to a double logarithm, which is also known to be optimal, thus we call this mechanism *OPT*^{2}. While OPT^2 also runs in polynomial time, it does have a higher computational cost than R2T in practice. Both R2T and OPT^2 are simple enough that they can be easily implemented on top of any RDBMS and an LP solver. Experimental results show that they offer order-of-magnitude improvements in terms of utility over existing techniques, even those specifically designed for graph pattern counting.

CCS Concepts: • **Information systems** → **Database query processing**; • **Security and privacy** → **Database and storage security**; • **Theory of computation** → **Theory of database privacy and security;**

Additional Key Words and Phrases: Differential privacy, SPJA query, foreign-key constraint

This work was supported by HKRGC under grants 16201318, 16201819, and 16205420; by an NTU-NAP startup grant; by the National Science Foundation under grant 2016393; and by DARPA and SPAWAR under contract N66001-15-C-4067.
Authors' Contact Information: Wei Dong, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore, Singapore; e-mail: wei_dong@ntu.edu.sg; Juanru Fang, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong; e-mail: jfangad@cse.ust.hk; Ke Yi, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong; e-mail: yike@cse.ust.hk; Yuchao Tao, Duke University, Durham, North Carolina, United States; e-mail: yuchao.tao@duke.edu; Ashwin Machanavajjhala, Computer Science, Duke University, Durham, North Carolina, United States; e-mail: ashwin@cs.duke.edu.

ACM 0362-5915/2024/11-ART13
https://doi.org/10.1145/3697831

ACM Reference Format:
Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2024. Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys. *ACM Trans. Datab. Syst.* 49, 4, Article 13 (November 2024), 40 pages. https://doi.org/10.1145/3697831

1 Introduction

Differential privacy (DP), already deployed by Apple [13], Google [27], Microsoft [14], and the U.S. Census Bureau [38], has become the standard notion for private data release, due to its strong protection of individual information. Informally speaking, DP requires indistinguishability of the query results whether any particular individual's data is included or not in the database. The standard *Laplace mechanism* first finds GS_Q, the *(global) sensitivity*, of the query—that is, how much the query result may change if an individual's data is added/removed from the database. Then it adds a Laplace noise calibrated accordingly to the query result to mask this difference. However, this mechanism runs into issues in a relational database, as illustrated in the following example.

Example 1.1. Consider a simple join-counting query

$$Q := |R_1(\underline{x_1}, \dots) \bowtie R_2(x_1, x_2, \dots)|.$$

Here, the underlined attribute $\underline{x_1}$ is the **primary key (PK)**, whereas $R_2.x_1$ is a **foreign key (FK)** referencing $R_1.x_1$. For instance, R_1 may store customer information where x_1 is the customer ID and R_2 stores the orders the customers have placed. Then this query simply returns the total number of orders; more meaningful queries could be formed with some predicates—for example, all customers from a certain region and/or orders in a certain category. Furthermore, suppose the customers, namely the tuples in R_1, are the entities whose privacy we aim to protect.

What is the GS_Q for this query? It is, unfortunately, ∞. This is because a customer, theoretically, could have an unbounded number of orders, and adding such a customer to the database can cause an unbounded change in the query result. A simple fix is to assume a finite GS_Q, which can be justified in practice because we may never have a customer with, say, more than a million orders. However, as assuming such a GS_Q limits the allowable database instances, one tends to be conservative and sets a large GS_Q. This allows the Laplace mechanism to work, but adding noise of this scale clearly eliminates any utility of the released query answer. □

1.1 The Truncation Mechanism

The preceding issue was first identified by Kotsogiannis et al. [34], who also formalized the *DP policy for relational databases with FK constraints.* The essence of their model (a rigorous definition is given in Section 3) is that the individuals and their private data are stored in separate relations that are linked by FKs. This is perhaps the most crucial feature of the relational model, yet it causes a major difficulty in designing DP mechanisms as illustrated previously. Their solution is the *truncation mechanism*, which simply deletes all customers with more than τ orders before applying the Laplace mechanism, for some threshold τ. After truncation, the query has sensitivity τ, so adding a noise of scale τ is sufficient.

Truncation is a special case of Lipschitz extensions and has been studied extensively for graph pattern counting queries [33] and machine learning [1]. A critical issue for the truncation mechanism is the bias-variance tradeoff: in one extreme $\tau = GS_Q$, it degenerates into the naive Laplace mechanism with a large noise (i.e., large variance); in the other extreme $\tau = 0$, the truncation introduces a bias as large as the query answer. The issue of how to choose a near-optimal τ has been extensively studied in the statistics and machine learning community [2, 3, 30, 39, 45]. A key challenge there is that the selection of τ must also be done in a DP manner. In fact, the particular query in Example 1.1 is equivalent to the one-dimensional mean (sum) estimation problem, which is a basic building block for many machine learning tasks like stochastic gradient descent [1, 7, 49] and clustering [50, 51].

1.2 The Issue with Self-Joins

While self-join-free queries are equivalent to mean (sum) estimation (see Section 4 for a more formal statement), which have been well studied, self-joins introduce another challenge unique to relational queries. In particular, all techniques from the statistics and machine learning literature [2, 3, 30, 39, 45] for choosing a τ critically rely on the fact that the individuals are independent (i.e., adding/removing one individual does not affect the data associated with another), which is not true when the query involves self-joins. In fact, when there are self-joins, even the truncation mechanism itself fails, as illustrated in the following example.

Example 1.2. Suppose we extend the query from Example 1.1 to the following one with a self-join:

$$Q := |R_1(\underline{x_1}, \dots,) \bowtie R_1(\underline{y_1}, \dots) \bowtie R_2(x_1, y_1, x_2, \dots)|.$$

Note that the PK of R_1 has been renamed differently in the two logical copies R_1 so that they join different attributes of R_2. For instance, R_2 may store the transactions between pairs of customers, and this query would count the total number of transactions. Again, predicates can be added to make the query more meaningful.

Let G be an undirected τ-regular graph (i.e., every vertex has degree τ) with n vertices. We will construct an instance $\mathbf{I} = (I_1, I_2)$, on which the truncation mechanism fails. Here, I_1, I_2 are instances corresponding to relation R_1 and R_2 in Example 1.2. Let I_1 be the vertices of G, and let I_2 be the edges (each edge will appear twice as G is undirected). Thus, Q simply returns the number of edges in the graph times 2. Let $\mathbf{I}'$ be the neighboring instance corresponding to G', to which we add a vertex v that connects to every existing vertex. Note that in G', v has degree n while every other vertex has degree $\tau+1$. Now truncating by τ fails DP: the query answer on $\mathbf{I}$ is $n\tau$, and that on $\mathbf{I}'$ is 0 (all vertices are truncated). Adding noise of scale τ cannot mask this gap, violating the DP definition. □

The reason the truncation mechanism fails is that the preceding underlined claim does not hold in the presence of self-joins. More fundamentally, this is due to the correlation among the individuals introduced by self-joins. In the preceding example, we see that the addition of one node may cause the degrees of many others to increase. For the problem of graph pattern counting under node-DP, which can be formulated as a multi-way self-join counting query on the special schema $\mathbf{R} = \{\texttt{Node}(\underline{\texttt{ID}}), \texttt{Edge}(\texttt{src}, \texttt{dst})\}$, Kasiviswanathan et al. [33] propose an LP-based truncation mechanism (to differentiate, we will call the preceding truncation mechanism *naive truncation*) to fix the issue, but they do not study how to choose τ. As a result, while their mechanism satisfies DP, there is no optimality guarantee in terms of utility. In fact, if τ is chosen inappropriately, their error can be even larger than GS_Q—namely, worse than the naive Laplace mechanism.

1.3 Our Contributions

We start by studying how to choose a near-optimal τ in a DP manner in the presence of self-joins. As with all prior τ-selection mechanisms over mean (sum) estimation [2, 3, 30, 39, 45] and self-join-free queries [52], we first assume that the *global sensitivity* of the given query Q is bounded by GS_Q. Since one tends to set a large GS_Q as argued in Example 1.1, we must try to minimize the dependency on GS_Q.

Our first contribution (Section 5) is a simple and general DP mechanism, called **Race-to-the-Top (R2T)**, which can be used to adaptively choose τ in combination with any valid DP truncation mechanism that satisfies certain properties. In fact, it does not choose τ per se; instead, it directly returns a privatized query answer with error at most $O\left(\log(GS_Q)\log\log(GS_Q)\cdot DS_Q(\mathbf{I})\right)$ for any instance $\mathbf{I}$ with constant probability. While we defer the formal definition of $DS_Q(\mathbf{I})$ to Section 4, what we can show is that it is an *per-instance lower bound*—that is, any valid DP mechanism has to incur error $\Omega\left(DS_Q(\mathbf{I})\right)$ on $\mathbf{I}$ (in a certain sense). Thus, the error of R2T is *instance-optimal* up to logarithmic factors in GS_Q.

However, as we see in Example 1.2, naive truncation is not a valid DP mechanism in the presence of self-joins. In Section 5.1, we extend the LP-based mechanism of Kasiviswanathan et al. [33], which only works for graph pattern counting queries, to general queries on an arbitrary relational schema that uses the four basic relational operators: selection (with arbitrary predicates), projection, join (including self-join), and sum aggregation. When plugged into R2T, this yields the first DP mechanism for answering arbitrary SPJA queries in a database with FK constraints. For SJA queries, the utility is instance-optimal, whereas the optimality guarantee for SPJA queries (Section 5.2) is slightly weaker, but we argue that this is unavoidable.

While R2T has been shown to achieve high utility and efficiency, two issues remain. The first is the assumption of a bounded GS_Q, which, as mentioned earlier, restricts the space of allowable database instances. The second issue is that its error is an $O(\log(GS_Q)\log\log(GS_Q))$-factor, called the *optimality ratio*, higher than the lower bound $DS_Q(I)$. It is not clear if this is the best one can achieve. In this extended article, we address these two issues by designing a new mechanism (Section 6) that achieves an error of $O\left(\log\log(DS_Q(\mathbf{I}))\cdot DS_Q(\mathbf{I})\right)$ on any instance $\mathbf{I}$ with constant probability,[1] without making any *a priori* assumptions on GS_Q. Note that $DS_Q(\mathbf{I})$ is smaller than GS_Q for any $\mathbf{I}$ (so this is an exponential improvement in the optimality ratio), whereas the latter can even be infinity if no restriction is put on the allowable instances. Very recently, such a doubly logarithmic optimality ratio has been shown to be the best possible even for self-join-free queries [22]. For this reason, we call the new mechanism OPT^2—namely, it is down-neighborhood-optimal with an optimal optimality ratio. We also extend OPT^2 to SPJA queries while maintaining an optimality guarantee similar to that of R2T, albeit somewhat weaker.

Despite their nontrivial utility analysis, the mechanisms of R2T and OPT^2 are actually very simple, and they can be built on top of any RDMBS and an LP solver. To demonstrate their practicality, we built a system prototype (Section 9) using PostgreSQL and CPLEX. Experimental results (Section 10) show they can provide order-of-magnitude improvements in terms of utility over the state-of-the-art DP-SQL engines. We obtain similar improvements even over node-DP mechanisms that are specifically designed for graph pattern counting problems, which are just special SJA queries. Furthermore, the experimental results show that while OPT^2 has better utility as indicated by the theory, it does incur a higher computational overhead (although still polynomial). In practice, the user may choose one of them depending on whether higher utility or higher efficiency is more desired.

R2T has been proposed in the conference version of this article [16]. In this article, we introduce OPT^2, which removes the assumption of a bounded GS_Q and achieves the optimal optimality ratio.

1.4 Organization

The article is organized as follows. After reviewing the related work in Section 2, we begin the technical development in Section 3. In Sections 4 and 5, we present R2T. In Section 6, we describe OPT^2. Section 7 gives a utility analysis for prior work [52], and Section 8 discusses our extension. Finally, Section 9 introduces the system implement, then Section 10 presents the experimental results. Section 11 provides additional discussion.

[1]All our derived error bounds hold with a high probability $1-\beta$ for any $\beta > 0$, whereas in Section 1, we state the results with a constant β for simplicity.

2 Related Work

Answering arbitrary SQL queries under DP is the holy grail of private query processing. Most early work focuses on answering a given set of counting queries over a single relation with different predicates (namely, SA queries with count aggregation) [6, 8, 12, 29, 36, 42, 47, 48, 55, 58]. Some works [8, 36, 42] design mechanisms that are optimal for the given query set, but over the *worst-case* database. In particular, if the set consists of just one query, their optimality degenerates into worst-case optimality.

Most existing work on join queries can only support restricted types of joins, such as PK-PK joins [4, 40, 41, 44, 46] and joins with a fixed join attribute [54]. A number of recent papers try to extend the support for joins, but as we see in Example 1.2, certain features like self-joins are tricky to handle correctly. PrivateSQL [34] uses naive truncation to truncate the tuples with high degree, so it does not really meet the DP requirement when there are self-joins. In a subsequent work, Tao et al. [52] use naive truncation to truncate the tuples with high sensitivity for self-join-free queries and they propose a mechanism to select τ. However, our analysis (see Section 7) shows that the error of their mechanism is $\Omega\left(GS_Q/\log(GS_Q)\right)$ with constant probability—that is, it is at most a logarithmic-factor better than the naive Laplace mechanism that adds noise of scale GS_Q. We reduce the dependency on GS_Q from (near) linear to logarithmic. We also compare with their mechanism experimentally for self-join-free queries in Section 10.

Smooth sensitivity [43] is a popular approach for dealing with self-joins. Elastic sensitivity [31] and residual sensitivity [19, 20], both of which are efficiently computable versions of smooth sensitivity, can handle self-joins correctly. However, as we argue in Section 4, smooth sensitivity (including any efficiently computable version) cannot support FK constraints, which are important to modeling the relationship between an individual and all of his/her associated data in a relational database. Consequently, they do not support node-DP for graph pattern counting, which is an important special case of FK constraints.

Node-DP and edge-DP are two popular DP policies for private graph data, which respectively are the special cases of having FK or no FK constraints in a relational database, as elaborated in Section 3.2. For node-DP, Kasiviswanathan et al. [33] combine naive truncation and smooth sensitivity, and also propose an LP-based truncation mechanism, whereas Blocki et al. [9] develop a smooth distance estimator. All these mechanisms require a τ given in advance. As such, none of them has any utility guarantees. Experimentally, we show in Section 10 (cf. Table 3) that the error of these mechanisms is highly sensitive to τ, while there is no fixed τ that works well for all queries and datasets. However, our mechanisms can always adaptively choose a τ that is provably close to the optimal one tuned (note that the tuning violates DP!) for each particular query/dataset. For edge-DP, better utility can be achieved [9, 32, 43, 57], but the privacy protection is weaker.

The recursive mechanism [11] also achieves an error close to $DS_Q(\mathbf{I})$, but without showing its instance optimality. More importantly, it adopts an approach that is complicated and different from the mainstream ones (e.g., the truncation mechanism and smooth sensitivity). In addition, its high computational costs prevent it from being used in practice. In our experiments, we were able to finish running this mechanism (with a time limit of 6 hours) only on the three test cases with the smallest query result size.

3 Preliminaries

3.1 Database Queries

Letting $\mathbf{R}$ be a database schema, we start with a multi-way join:

$$J := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n),$$

where $R_1, \ldots, R_n$ are relation names in $\mathbf{R}$ and each $\mathbf{x}_i$ is a set of $arity(R_i)$ variables, where $arity(R_i)$ is the number of attributes in R_i. When considering self-joins, there can be repeats (i.e., $R_i = R_j$); in this case, we must have $\mathbf{x}_i \neq \mathbf{x}_j$, or one of the two atoms will be redundant. Let $var(J) := \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$.

Let $\mathbf{I}$ be a database instance over $\mathbf{R}$. For any $R \in \mathbf{R}$, denote the corresponding relation instance in $\mathbf{I}$ as $\mathbf{I}(R)$. This is a *physical relation instance* of R. We use $\mathbf{I}(R, \mathbf{x})$ to denote $\mathbf{I}(R)$ after renaming its attributes to $\mathbf{x}$, which is also called a *logical relation instance* of R. When there are self-joins, one physical relation instance may have multiple logical relation instances; they have the same rows but with different column (variable) names.

A JA or SJA query Q aggregates over the join results $J(\mathbf{I})$. More abstractly, let $\psi : \mathbf{dom}(var(J)) \rightarrow \mathbb{N}$ be a function that assigns non-negative integer weights to the join results, where $\mathbf{dom}(var(J))$ denotes the domain of $var(J)$. The result of evaluating Q on $\mathbf{I}$ is

$$Q(\mathbf{I}) := \sum_{q \in J(\mathbf{I})} \psi(q). \tag{1}$$

Note that the function ψ only depends on the query. For a counting query, $\psi(\cdot) \equiv 1$; for an aggregation query, for example, $\texttt{SUM}(A * B)$, $\psi(q)$ is the value of $A * B$ for q. An SJA query with an arbitrary predicate over $var(J)$ can be easily incorporated into this formulation: if some $q \in J(\mathbf{I})$ does not satisfy the predicate, we simply set $\psi(q) = 0$.

Example 3.1. Graph pattern counting queries can be formulated as SJA queries. Suppose we store a graph in a relational database by the schema $\mathbf{R} = \{\texttt{Node}(\underline{\texttt{ID}}), \texttt{Edge}(\texttt{src}, \texttt{dst})\}$, where `src` and `dst` are FKs referencing `ID`, then the number of length-3 paths can be counted by first computing the join

$$\texttt{Edge}(\texttt{A}, \texttt{B}) \bowtie \texttt{Edge}(\texttt{B}, \texttt{C}) \bowtie \texttt{Edge}(\texttt{C}, \texttt{D}),$$

followed by a count aggregation. Note that this also counts triangles and non-simple paths (e.g., x-y-x-z), which may or may not be considered as length-3 paths depending on the application. If not, they can be excluded by introducing a predicate (i.e., redefining ψ) $\texttt{A} \neq \texttt{C} \wedge \texttt{A} \neq \texttt{D} \wedge \texttt{B} \neq \texttt{D}$. If the graph is undirected, then the query counts every path twice, so we should divide the answer by 2. Alternatively, we may introduce the predicate $\texttt{A} < \texttt{D}$ to eliminate the double counting. □

Finally, for an SPJA query where the output variables are $\mathbf{y} \subset var(J)$, we simply replace $J(\mathbf{I})$ with $\pi_{\mathbf{y}} J(\mathbf{I})$ in (1). Note that we use the relational algebra semantics of a projection, where duplicates are removed. If not, the projection would not make any difference in the aggregate. In fact, it is precisely the duplicate removal that makes SPJA queries more difficult than SJA queries in terms of optimality, as we argue in Section 5.2.

3.2 DP in Relational Databases with FK Constraints

We adopt the DP policy in the work of Kotsogiannis et al. [34], which defines neighboring instances by taking FK constraints into consideration. We model all FK relationships as a directed acyclic graph over $\mathbf{R}$ by adding a directed edge from R to R' if R has an FK referencing the PK of R'. There is a[2] designated *primary private relation* R_P, and any relation that has a direct or indirect FK referencing R_P is called a *secondary private relation*. The *referencing* relationship over the tuples is defined recursively as follows: (1) any tuple $t_P \in \mathbf{I}(R_P)$ said to reference itself; (2) for $t_P \in \mathbf{I}(R_P)$, $t \in \mathbf{I}(R)$, $t' \in \mathbf{I}(R')$, if t' references t_P, R has an FK referencing the PK of R', and the FK of t equals to the PK of t', then we say that t references t_P.

[2]For most parts of the article, we consider the case where there is only one *primary private relation* in $\mathbf{R}$; the case with multiple primary private relations is discussed in Section 8.

For a join result $q \in J(\mathbf{I})$, we say that q references $t_P \in \mathbf{I}(R_P)$ if $t_P \bowtie q \neq \emptyset$. Let $N = |\mathbf{I}(R_P)|$ and $M = |J(\mathbf{I})|$. Let $[k] = \{1, 2, \ldots, k\}$. For $i \in [N]$, let $t_i(\mathbf{I})$ be the ith tuple in $\mathbf{I}(R_P)$; for $j \in [M]$, let $q_j(\mathbf{I})$ be the jth join result in $J(\mathbf{I})$. To describe the relationships between tuples and join results, we use $C_i(\mathbf{I})$ and $D_j(\mathbf{I})$ to denote (the indices of) the set of join results that reference $t_i(\mathbf{I})$ and the set of tuples that $q_j(\mathbf{I})$ references—that is,

$$C_i(\mathbf{I}) = \{j : q_j(\mathbf{I}) \text{references} t_i(\mathbf{I})\}, \tag{2}$$

$$D_j(\mathbf{I}) = \{i : q_j(\mathbf{I}) \text{references} t_i(\mathbf{I})\}. \tag{3}$$

Two instances $\mathbf{I}$ and $\mathbf{I}'$ are considered neighbors if $\mathbf{I}'$ can be obtained from $\mathbf{I}$ by deleting a tuple t_P and all tuples referencing it. This ensures that the FK constraints are preserved. We use the notation $\mathbf{I} \sim \mathbf{I}'$ to denote two neighboring instances, and $\mathbf{I} \sim_{t_P} \mathbf{I}'$ denotes that all tuples in the difference between $\mathbf{I}$ and $\mathbf{I}'$ reference the tuple $t_P \in R_P$. We write $\mathbf{I}' \subseteq \mathbf{I}$ if $\mathbf{I}'$ can be obtained from $\mathbf{I}$ by removing a set of tuples $\{t_P : t_P \in \mathbf{I}(R_P)\}$ and all tuples referencing them. We thus have $\mathbf{I}'(R) \subseteq \mathbf{I}(R)$ for any $R \in \mathbf{R}$.

Example 3.2. Consider the TPC-H schema:

$$\mathbf{R} = \{\texttt{Nation}(\underline{\texttt{NK}}), \texttt{Customer}(\underline{\texttt{CK}}, \texttt{NK}), \texttt{Order}(\underline{\texttt{OK}}, \texttt{CK}), \texttt{Lineitem}(\texttt{OK})\}.$$

If the customers are the individuals whose privacy we wish to protect, then we designate `Customer` as the primary private relation, which implies that `Order` and `Lineitem` will be secondary private relations, whereas `Nation` will be public. Note that once `Customer` is designated as a primary private relation, the information in `Order` and `Lineitem` is also protected since the privacy induced by `Customer` is stronger than that induced by `Order` and `Lineitem`. Alternatively, one may designate `Order` as the primary private relation, which implies that `Lineitem` will be a secondary private relation, whereas `Customer` and `Nation` will be public. This would result in weaker privacy protection but offer higher utility. This is because each individual corresponds to fewer join results, thereby necessitating less noise injection to maintain privacy. □

Some queries, as given in Example 3.1, may be *incomplete*—that is, it has a variable that is an FK but its referenced PK does not appear in the query Q. Following Kotsogiannis et al. [34], we always make the query complete by iteratively adding those relations whose PKs are referenced to Q. The PKs will be given variables names matching the FKs. For example, for the query in Example 3.1, we add `Node(A)`, `Node(B)`, `Node(C)`, and `Node(D)`.

The preceding DP policy incorporates both edge-DP and node-DP, two commonly used DP policies for private graph analysis, as special cases. In Example 3.1, by designating `Edge` as the private relation (`Node` is thus public, and we may even assume it contains all possible vertex IDs), we obtain edge-DP; for node-DP, we add FK constraints from `src` and `dst` to `ID`, and designate `Node` as the primary private relation, whereas `Edge` becomes a secondary private relation.

Definition 3.3 (Differential Privacy). For $\varepsilon > 0$, a mechanism M is ε-DP if for any neighboring instances $\mathbf{I} \sim \mathbf{I}'$ and any output y,

$$\Pr[M(\mathbf{I}) = y] \leq e^{\varepsilon} \cdot \Pr[M(\mathbf{I}') = y].$$

Typical values of ε used in practice range from 0.1 to 10, where a smaller value corresponds to stronger privacy protection.

3.3 Common DP Mechanisms

The standard DP mechanism is the Laplace mechanism [24]. Let $Lap(b)$ denote a random variable drawn from the Laplace distribution with scale b and $GS_Q = \max_{\mathbf{I} \sim \mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')|$ be the *global sensitivity* of Q.

ALGORITHM 1: SVT

Input: $\mathbf{I}$, T, ε, $Q_1(\mathbf{I}), Q_2(\mathbf{I}), \ldots$

$\tilde{T} \leftarrow T + Lap(2/\varepsilon)$;
for $\ell \leftarrow 1, 2, \ldots$ **do**
  $\tilde{Q}_\ell(\mathbf{I}) \leftarrow Q_\ell(\mathbf{I}) + Lap(4/\varepsilon)$;
  if $\tilde{Q}_\ell(\mathbf{I}) > \tilde{T}$ **then**
    Break;
  end
end
return ℓ;

Lemma 3.4. *The Laplace mechanism $M(\mathbf{I}) = Q(\mathbf{I}) + Lap(GS_Q/\varepsilon)$ preserves ε-DP.*

The **sparse vector technique (SVT)** [25] has as input a (possibly infinite) sequence of queries, $Q_1(\mathbf{I}), Q_2(\mathbf{I}), \ldots$, where each has global sensitivity 1, and a threshold T. It targets to find the first query whose answer is above T. The detailed algorithm is given in Algorithm 1.

Lemma 3.5 ([22]). *The SVT preserves ε-DP. If there exists a k such that $Q_k(\mathbf{I}) \geq T + 6\ln(2/\beta)/\varepsilon$, then with probability at least $1-\beta$, SVT returns an $\ell \leq k$ such that $Q_\ell(\mathbf{I}) \geq T - 6\ln(2k/\beta)/\varepsilon$.*

4 Instance Optimality of DP Mechanisms with FK Constraints

Global Sensitivity and Worst-Case Optimality. The Laplace mechanism adds noise calibrated to GS_Q to the query answer. However, either a join or a sum aggregation makes GS_Q unbounded. The issue with the former is illustrated in Example 1.1, where a customer may have unbounded orders; a sum aggregation with an unbounded ψ results in the same situation. Thus, as with prior work [2, 3, 30, 39, 45, 52], we first restrict to a set of instances $\mathcal{I}$ such that

$$\max_{\mathbf{I}\in\mathcal{I}, \mathbf{I}'\in\mathcal{I}, \mathbf{I}\sim\mathbf{I}'} |Q(\mathbf{I}) - Q(\mathbf{I}')| = GS_Q, \tag{4}$$

where GS_Q is a parameter given in advance. For the query in Example 1.1, this is equivalent to assuming that a customer is allowed to have at most GS_Q orders in any instance. We will remove this assumption in Section 6.

For general queries, the situation is more complicated. We first consider SJA queries. Given an instance $\mathbf{I}$ and an SJA query Q, for a tuple $t_P \in \mathbf{I}(R_P)$, its *sensitivity* is

$$S_Q(\mathbf{I}, t_P) := \sum_{q\in J(\mathbf{I})} \psi(q)\mathbb{I}(q\mathrm{references}t_P), \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function. For SJA queries, (4) is equivalent to

$$\max_{\mathbf{I}\in\mathcal{I}} \max_{t_P\in\mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P) = GS_Q.$$

For self-join-free SJA queries, it is clear that

$$Q(\mathbf{I}) = \sum_{t_P\in R_P} S_Q(\mathbf{I}, t_P),$$

which turns the problem into a sum estimation problem. However, when self-joins are present, this equality no longer holds since one join result q references multiple t_P's. This also implies that removing one tuple from $\mathbf{I}(R_P)$ may affect multiple $S_Q(\mathbf{I}, t_P)$'s, making the neighboring relationship more complicated than in the sum estimation problem, where two neighboring instances differ by only one datum [2, 3, 30, 39, 45].

What notion of optimality shall we use for DP mechanisms over SJA queries? The traditional worst-case optimality is meaningless, since the naive Laplace mechanism that adds noise of scale GS_Q is already worst-case optimal, just by the definition of GS_Q. In fact, the basis of the entire line of work on the truncation mechanism and smooth sensitivity is the observation that typical instances should be much easier than the worst case, so these mechanisms all add instance-specific noises, which are often much smaller than the worst-case noise level GS_Q.

Instance Optimality. The standard notion of optimality for measuring the performance of an algorithm on a per-instance basis is *instance optimality*. More precisely, let $\mathcal{M}$ be the class of DP mechanisms and let[3]

$$\mathcal{L}_{\text{ins}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \min\{\xi : \Pr[|M'(\mathbf{I}) - Q(\mathbf{I})| \leq \xi] \geq 2/3\}$$

be the lower bound any $M' \in \mathcal{M}$ can achieve (with probability $2/3$) on $\mathbf{I}$, then the standard definition of instance optimality requires us to design an M such that

$$\Pr[|M(\mathbf{I}) - Q(\mathbf{I})| \leq c \cdot \mathcal{L}_{\text{ins}}(\mathbf{I})] \geq 2/3 \tag{6}$$

for every $\mathbf{I}$, where c is called the *optimality ratio*. Unfortunately, for any $\mathbf{I}$, one can design a trivial $M'(\cdot) \equiv Q(\mathbf{I})$ that has 0 error on $\mathbf{I}$ (but fails miserably on other instances), so $\mathcal{L}_{\text{ins}}(\cdot) \equiv 0$, which rules out instance-optimal DP mechanisms by a standard argument [26].

To avoid such a trivial M', Asi and Duchi [5] and Dong and Yi [20] consider a relaxed version of instance optimality where we compare M against any M' that is required to work well not just on $\mathbf{I}$ but also on its neighbors—that is, we raise the target error from $\mathcal{L}_{\text{ins}}(\mathbf{I})$ to

$$\mathcal{L}_{\text{nbr}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \max_{\mathbf{I}' \in \mathcal{I}, \mathbf{I} \sim \mathbf{I}'} \min\{\xi : \Pr[|M'(\mathbf{I}') - Q(\mathbf{I}')| \leq \xi] \geq 2/3\}.$$

Vadhan [53] observes that $\mathcal{L}_{\text{nbr}}(\mathbf{I}) \geq LS_Q(\mathbf{I})/2$, where

$$LS_Q(\mathbf{I}) := \max_{\mathbf{I}' \in \mathcal{I}, \mathbf{I}' \sim \mathbf{I}} |Q(\mathbf{I}) - Q(\mathbf{I}')|$$

is the *local sensitivity* of Q at $\mathbf{I}$. This instance optimality has been used for certain machine learning problems [5] and conjunctive queries without FKs [20]. However, it has an issue for SJA queries in a database with FK constraints: for any $\mathbf{I}$, we can add a t_P to $\mathbf{I}(R_P)$ together with tuples in the secondary private relations all referencing t_P, obtaining an $\mathbf{I}'$ such that $S_Q(\mathbf{I}', t_P) = GS_Q$ (i.e., $LS_Q(\cdot) \equiv GS_Q$). This means that this relaxed instance optimality degenerates into worst-case optimality. This is also why smooth sensitivity, including all its efficiently computable versions [19, 20, 31, 43], will not have better utility than the naive Laplace mechanism on databases with FK constraints, since they are all no lower than the local sensitivity.

The reason the preceding relaxation is "too much" is that we require M' to work well on any neighbor $\mathbf{I}'$ of $\mathbf{I}$. Under the neighborhood definition with FK constraints, this means that $\mathbf{I}'$ can be any instance obtained from $\mathbf{I}$ by adding a tuple t_P and *arbitrary* tuples referencing t_P in the secondary private relations. This is too high a requirement for M', hence too low an optimality notion for M.

To address the issue, Huang et al. [30] restrict the neighborhood in which M' is required to work well, but their definition only works for the mean estimation problem. For SJA queries under FK constraints, we revise $\mathcal{L}_{\text{nbr}}(\cdot)$ to

$$\mathcal{L}_{\text{d-nbr}}(\mathbf{I}) := \min_{M' \in \mathcal{M}} \max_{\mathbf{I}' : \mathbf{I} \sim \mathbf{I}', \mathbf{I}' \subseteq \mathbf{I}} \min\{\xi : \Pr[|M'(\mathbf{I}') - Q(\mathbf{I}')| \leq \xi] \geq 2/3\}.$$

[3]The probability constant $2/3$ can be changed to any constant larger than $1/2$ without affecting the asymptotics.

In other words, we require M' to work well only on $\mathbf{I}'$ and its *down-neighbors*, which can be obtained only by removing a tuple t_P already in $\mathbf{I}(R_P)$ and all tuples referencing t_P. Correspondingly, an instance-optimal M (w.r.t. the down-neighborhood) is one such that (6) holds where $\mathcal{L}_{\text{ins}}$ is replaced by $\mathcal{L}_{\text{d-nbr}}$.

Clearly, the smaller the neighborhood, the stronger the optimality notion. Our instance optimality notion is thus stronger than those in other works [5, 20, 30]. Note that for such an instance-optimal M (by our definition), there still exist $\mathbf{I}, M'$ such that M' does better on $\mathbf{I}$ than M, but if this happens, M' must do worse on one of the down-neighbors of $\mathbf{I}$, which is as typical as $\mathbf{I}$ itself.

Using the same argument from Vadhan [53], we have $\mathcal{L}_{\text{d-nbr}}(\mathbf{I}) \geq DS_Q(\mathbf{I})/2$, where

$$DS_Q(\mathbf{I}) := \max_{\mathbf{I}':\mathbf{I}\sim\mathbf{I}', \mathbf{I}'\subseteq\mathbf{I}} |Q(\mathbf{I}) - Q(\mathbf{I}')| = \max_{t_P\in\mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P) \tag{7}$$

is the *downward local sensitivity* of $\mathbf{I}$. Thus, $DS_Q(\mathbf{I})$ is a per-instance lower bound, which can be used to replace $\mathcal{L}_{\text{inc}}(\mathbf{I})$ in (6) in the definition of instance-optimal DP mechanisms.

5 R2T: Instance-Optimal Truncation

Our instance-optimal truncation mechanism, R2T, can be used in combination with any truncation method $Q(\mathbf{I}, \tau)$, which is a function $Q : \mathcal{I} \times \mathbb{N} \to \mathbb{N}$ with the following properties:

(1) For any τ, the global sensitivity of $Q(\cdot, \tau)$ is at most τ.
(2) For any τ, $Q(\mathbf{I}, \tau) \leq Q(\mathbf{I})$.
(3) For any $\mathbf{I}$, there exists a non-negative integer $\tau^*(\mathbf{I}) \leq GS_Q$ such that for any $\tau \geq \tau^*(\mathbf{I})$, $Q(\mathbf{I}, \tau) = Q(\mathbf{I})$.

Notice that the naive truncation for self-join-free SJA queries, as mentioned in Section 1.1, also adheres to these three properties. We will describe various choices for $Q(\mathbf{I}, \tau)$ depending on whether the query contains self-joins and/or projections in the subsequent sections—that is, self-join SJA queries in Section 5.1 and SPJA queries in Section 5.2.

Intuitively, such a $Q(\mathbf{I}, \tau)$ gives a stable (property (1)) underestimate (property (2)) of $Q(\mathbf{I})$, whereas it reaches $Q(\mathbf{I})$ for a sufficiently large τ (property (3)). Note that $Q(\mathbf{I}, \tau)$ itself is not DP. To make it DP, we can add $Lap(\tau/\varepsilon)$, which would turn it into an ε-DP mechanism by property (1). The issue, of course, is how to set τ. The basic idea of R2T is to try geometrically increasing values of τ and somehow pick the "winner" of the race.

Assuming such a $Q(\mathbf{I}, \tau)$, R2T works as follows. For a probability[4] β, we first compute[5]

$$\tilde{Q}(\mathbf{I}, \tau^{(i)}) := Q(\mathbf{I}, \tau^{(i)}) + Lap\left(\log(GS_Q)\frac{\tau^{(i)}}{\varepsilon}\right) - \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(i)}}{\varepsilon}, \tag{8}$$

for $\tau^{(i)} = 2^i$, $i = 1, \ldots, \log(GS_Q)$. Then R2T outputs

$$\tilde{Q}(\mathbf{I}) := \max\left\{\max_i \tilde{Q}(\mathbf{I}, \tau^{(i)}), Q(\mathbf{I}, 0)\right\}. \tag{9}$$

The privacy of R2T is straightforward: since $Q(\mathbf{I}, \tau^{(i)})$ has global sensitivity at most $\tau^{(i)}$, and the third term of (8) is independent of $\mathbf{I}$, each $\tilde{Q}(\mathbf{I}, \tau^{(i)})$ satisfies $\varepsilon/\log(GS_Q)$-DP by Lemma 3.4. Collectively, all $\tilde{Q}(\mathbf{I}, \tau^{(i)})$'s satisfy ε-DP by the basic composition theorem [26]. Finally, returning the maximum preserves DP by the post-processing property of DP.

[4] The probability β only concerns the utility, not privacy.
[5] The log has base 2 and ln has base e.

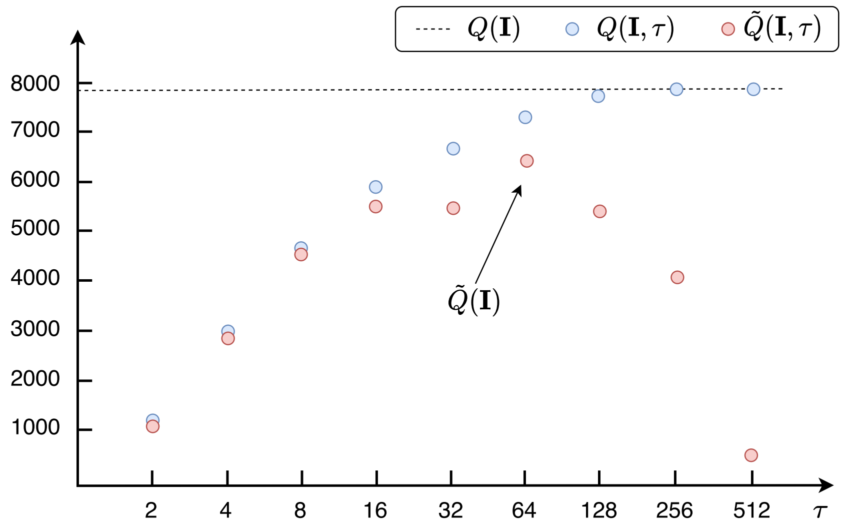

Fig. 1. An illustration of R2T.

Utility Analysis. For some intuition on why R2T offers good utility, please see Figure 1. By property (2) and property (3), as we increase τ, $Q(\mathbf{I},\tau)$ gradually approaches the true answer $Q(\mathbf{I})$ from below and reaches $Q(\mathbf{I},\tau) = Q(\mathbf{I})$ when $\tau \geq \tau^*(\mathbf{I})$. However, we cannot use $Q(\mathbf{I},\tau)$ or $\tau^*(\mathbf{I})$ directly, as this would violate DP. Instead, we only get to see $\tilde{Q}(\mathbf{I},\tau)$, which is masked with the noise of scale proportional to τ. We thus face a dilemma, that the closer we get to $Q(\mathbf{I})$, the more uncertain we are about the estimate $\tilde{Q}(\mathbf{I},\tau)$. To get out of the dilemma, we shift $Q(\mathbf{I},\tau)$ down by an amount that equals to the scale of the noise (if ignoring the log log factor). This penalty for $\tilde{Q}(\mathbf{I},\hat{\tau})$, where $\hat{\tau}$ is the smallest power of 2 above $\tau^*(\mathbf{I})$, will be on the same order as $\tau^*(\mathbf{I})$, so it will not affect its error by more than a constant factor, whereas taking the maximum ensures that the winner is at least as good as $\tilde{Q}(\mathbf{I},\hat{\tau})$. Meanwhile, the extra log log factor ensures that no $\tilde{Q}(\mathbf{I},\tau)$ overshoots the target. Next, we formalize the intuition.

Theorem 5.1. *On any instance* $\mathbf{I}$*, with probability at least* $1-\beta$*, we have*

$$Q(\mathbf{I}) - 4\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\frac{\tau^*(\mathbf{I})}{\varepsilon} \leq \tilde{Q}(\mathbf{I}) \leq Q(\mathbf{I}).$$

Proof. It suffices to show that each inequality holds with probability at least $1-\frac{\beta}{2}$. For the second inequality, since $Q(\mathbf{I},0) \leq Q(\mathbf{I})$, we just need to show that $\max_i \tilde{Q}(\mathbf{I},\tau^{(i)}) \leq Q(\mathbf{I})$. By a union bound, it suffices to show that $\tilde{Q}(\mathbf{I},\tau) \leq Q(\mathbf{I})$ with probability at most $\beta/(2\log(GS_Q))$ for each τ. This easily follows from property (2) of $Q(\mathbf{I},\tau)$ and the tail bound of the Laplace distribution:

$$\begin{aligned}
&\Pr[\tilde{Q}(\mathbf{I},\tau) > Q(\mathbf{I})]\\
\leq &\Pr[\tilde{Q}(\mathbf{I},\tau) > Q(\mathbf{I},\tau)]\\
= &\Pr\left[Lap\left(\log(GS_Q)\frac{\tau}{\varepsilon}\right) > \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau}{\varepsilon}\right]\\
= &\frac{\beta}{2\log(GS_Q)}.
\end{aligned}$$

For the first inequality, we discuss two cases $\tau^*(\mathbf{I}) = 0$ and $\tau^*(\mathbf{I}) \in (2^{i-1}, 2^i]$ for some $i \geq 1$. For the first case, by property (3) of $Q(\mathbf{I}, \tau)$, $Q(\mathbf{I}, 0) = Q(\mathbf{I})$. Therefore, $\tilde{Q}(\mathbf{I}) \geq Q(\mathbf{I}, 0) = Q(\mathbf{I})$. In the following, we discuss the second case where $\tau^*(\mathbf{I}) \in (2^{i-1}, 2^i]$. Note that $2^i \leq 2\tau^*(\mathbf{I})$. Let $\hat{\tau} = 2^i$. By the tail bound on the Laplace distribution, with probability at least $1 - \frac{\beta}{2}$, we have

$$\begin{aligned}\tilde{Q}(\mathbf{I}, \hat{\tau}) \geq & Q(\mathbf{I}, 2^i) - 2\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\frac{2^i}{\varepsilon} \\ = & Q(\mathbf{I}) - 2\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\frac{2^i}{\varepsilon} \quad (10) \\ \geq & Q(\mathbf{I}) - 4\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\frac{\tau^*(\mathbf{I})}{\varepsilon}. \quad (11)\end{aligned}$$

Note that (10) follows the third property of $Q(\mathbf{I}, \tau)$, and (11) is because $2^i \leq 2\tau^*(\mathbf{I})$. Finally, since $\tilde{Q}(\mathbf{I}) \geq \max_j \tilde{Q}(\mathbf{I}, \tau^{(i)}) \geq \tilde{Q}(\mathbf{I}, \hat{\tau})$, the first inequality also holds with probability at least $1 - \frac{\beta}{2}$. □

5.1 Truncation for SJA Queries

In this section, we design a $Q(\mathbf{I}, \tau)$ with $\tau^*(\mathbf{I}) = DS_Q(\mathbf{I})$ for SJA queries. Plugged into Theorem 5.1 with $\beta = 1/3$ and the definition of instance optimality, this turns R2T into an instance-optimal DP mechanism with an optimality ratio of $O\left(\log(GS_Q)\log\log(GS_Q)/\varepsilon\right)$.

For self-join-free SJA queries, each join result $q \in J(\mathbf{I})$ references only one tuple in R_P. Thus, the tuples in R_P are independent—that is, removing one does not affect the sensitivities of others. This means that naive truncation (i.e., removing all $S_Q(\mathbf{I}, t_P) > \tau$ and then summing up the rest) is a valid $Q(\mathbf{I}, \tau)$ that satisfies the three properties required by R2T with $\tau^*(\mathbf{I}) = DS_Q(\mathbf{I})$.

When there are self-joins, naive truncation does not satisfy property (1), as illustrated in Example 1.2, where all $S_Q(\mathbf{I}, t_P)$'s in two neighboring instances may differ. In the following, we generalize the LP-based mechanism for graph pattern counting [33] to arbitrary SJA queries and show that it satisfies the three properties with $\tau^*(\mathbf{I}) = DS_Q(\mathbf{I})$.

Given a SJA query Q and instance $\mathbf{I}$, recall that $Q(\mathbf{I}) = \sum_{q\in J(\mathbf{I})} \psi(q)$ and $C_i(\mathbf{I})$ is the indices of the set of join results referencing $t_i(\mathbf{I})$. For each $j \in [M]$, introduce a variable u_j, which represents the weight assigned to the join result $q_j(\mathbf{I})$. We return the optimal solution of the following LP as $Q(\mathbf{I}, \tau)$:

$$\begin{aligned} \text{maximize} \quad & Q(\mathbf{I}, \tau) = \sum_{j\in[M]} u_j, \\ \text{subject to} \quad & \sum_{j\in C_i(\mathbf{I})} u_j \leq \tau, \qquad i \in [N], \\ & 0 \leq u_j \leq \psi(q_j(\mathbf{I})), j \in [M]. \end{aligned}$$

Example 5.2. We now give a step-by-step example to show how this truncation method works together with R2T. Consider the problem of edge counting under node-DP, which corresponds to the SJA query

$$Q := |\sigma_{\mathtt{ID1}<\mathtt{ID2}}(\mathtt{Node(ID1)} \bowtie \mathtt{Node(ID2)} \bowtie \mathtt{Edge(ID1, ID2)})|$$

on the graph data schema introduced in Example 3.1. Note that in SQL, the query would be written as follows.

```
SELECT count(*) FROM Node AS Node1, Node AS Node2, Edge
WHERE Edge.src = Node1.ID AND Edge.dst = Node2.ID AND Node1.ID < Node2.ID
```

Suppose we set $GS_Q = 2^{10} = 1024$. For this particular Q, this means the maximum degree of any node in any instance $\mathbf{I} \in \mathcal{I}$ is 1024. We set $\beta = 0.1$ and $\varepsilon = 1$.

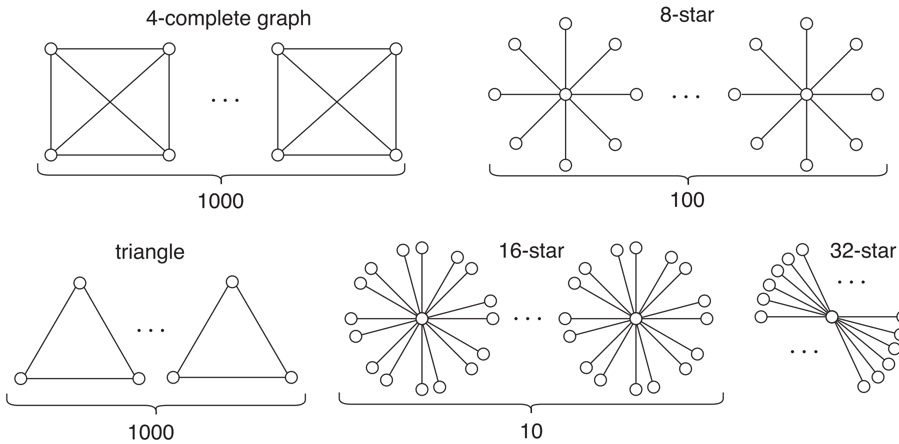

Fig. 2. Example of edge counting.

Now, suppose we are given an $\mathbf{I}$ containing 8103 nodes, which form 1000 triangles, 1000 4-cliques, 100 8-stars, 10 16-stars, and 1 32-star as shown in Figure 2. The true query result is

$$Q(\mathbf{I}) = 3 \times 1000 + 6 \times 1000 + 8 \times 100 + 16 \times 10 + 32 = 9992.$$

We run R2T with $\tau^{(i)} = 2^i$ for $i = 1, \dots, 8$. For each $\tau = \tau^{(i)}$, we assign a weight $u_j \in [0, 1]$ to each join result (i.e., an edge) that satisfies the predicate ID1 < ID2. To calculate $Q(\mathbf{I}, \tau)$, we can consider the LP on each clique/star separately. For a triangle, the optimal LP solution always assigns $u_j = 1$ for each edge. For each 4-clique, it assigns $2/3$ to each edge for $\tau = 2$ and 1 for $\tau \geq 4$. For each k-star, the LP optimal solution is $\min(k, \tau)$. Thus, the optimal LP solutions are

$$\begin{aligned}
Q(\mathbf{I}, 2) &= 1 \times 3000 + \frac{2}{3} \times 6000 + 2 \times 100 + 2 \times 10 + 2 \times 1 = 7222,\\
Q(\mathbf{I}, 4) &= 1 \times 3000 + 1 \times 6000 + 4 \times 100 + 4 \times 10 + 4 \times 1 = 9444,\\
Q(\mathbf{I}, 8) &= 1 \times 3000 + 1 \times 6000 + 8 \times 100 + 8 \times 10 + 8 \times 1 = 9888,\\
Q(\mathbf{I}, 16) &= 1 \times 3000 + 1 \times 6000 + 8 \times 100 + 16 \times 10 + 16 \times 1 = 9976.
\end{aligned}$$

In addition, we have $Q(\mathbf{I}, 0) = 0$ and $Q(\mathbf{I}, \tau) = 9992$ for $\tau \geq 32$.

Then, let us see how to run R2T with these $Q(\mathbf{I}, \tau)$'s. For concreteness, assume $Lap(1)$ returns -1 and 1 in turn. Plugging these into (8), we have the following:

$$\begin{aligned}
\tilde{Q}(\mathbf{I}, 2) =& 7222 + (-1) \cdot 20 - 92.1 = 7109.9,\\
\tilde{Q}(\mathbf{I}, 4) =& 9444 + 1 \cdot 40 - 184 = 9300,\\
\tilde{Q}(\mathbf{I}, 8) =& 9888 + (-1) \cdot 80 - 368 = 9440,\\
\tilde{Q}(\mathbf{I}, 16) =& 9976 + 1 \cdot 160 - 737 = 9399,\\
\tilde{Q}(\mathbf{I}, 32) =& 9992 + (-1) \cdot 320 - 1474 = 8198,\\
\tilde{Q}(\mathbf{I}, 64) =& 9992 + 1 \cdot 640 - 2947 = 7685,\\
&\dots
\end{aligned}$$

Finally, with (9), we have $\tilde{Q}(\mathbf{I}) = \tilde{Q}(\mathbf{I}, 8) = 9440$. □

Utility Analysis. The utility guarantee of the R2T instantiation for SJA queries follows from the following lemma.

LEMMA 5.3. *For SJA queries, the $Q(\mathbf{I}, \tau)$ defined previously satisfies the three properties required by R2T with $\tau^*(\mathbf{I}) = DS_Q(\mathbf{I})$.*

PROOF. Property (2) easily follows from the constraint $u_j \leq \psi(q_j(\mathbf{I}))$. For property (3), observe that for SJA queries, for any $i \in [N]$, $S_Q(\mathbf{I}, t_i(\mathbf{I})) = \sum_{j \in C_i(\mathbf{I})} \psi(q_j(\mathbf{I}))$. So when $\tau \geq DS_Q(\mathbf{I})$, all constraints $\sum_{j \in C_i(\mathbf{I})} u_j \leq \tau$ are satisfied automatically and we can set $u_j = \psi(q_j(\mathbf{I}))$ for all j.

In the following, we prove property (1)—that is, for any $\mathbf{I} \sim \mathbf{I}'$, $Q(\mathbf{I}, \tau)$ and $Q(\mathbf{I}', \tau)$ differ by at most τ. Without loss of generality, assume $\mathbf{I} \subseteq \mathbf{I}'$. It is clear that $J(\mathbf{I}) \subseteq J(\mathbf{I}')$, and we order the join results in $J(\mathbf{I}')$ in such a way that the extra join results are at the end. This means that the two LPs on $\mathbf{I}$ and $\mathbf{I}'$ share common variables $u_1, \ldots, u_M$, whereas the latter has some extra variables $u_{M+1}, \ldots, u_{M'}$. Each constraint $\sum_{j \in C_i(\mathbf{I})} u_j \leq \tau$ in the LP on $\mathbf{I}$ has a counterpart $\sum_{j \in C_i(\mathbf{I}')} u_j \leq \tau$ in the LP on $\mathbf{I}'$, where $C_i(\mathbf{I}) \subseteq C_i(\mathbf{I}')$. Let t_{i^*} be the tuple in $\mathbf{I}'(R_P)$ that all tuples in $\mathbf{I}' - \mathbf{I}$ reference. Note that t_{i^*} may or may not appear in $\mathbf{I}$. But in either case, the LP on $\mathbf{I}'$ has a constraint $\sum_{j \in C_{i^*}(\mathbf{I}')} u_j \leq \tau$ and $C_{i^*}(\mathbf{I}')$ contains all extra variables in the LP on $\mathbf{I}'$.

Let $\{u_j^*(\mathbf{I})\}_j$ be the optimal solution of the LP on $\mathbf{I}$. We extend it to a solution $\{u_j(\mathbf{I}')\}_j$ of the LP on $\mathbf{I}'$, by setting $u_j(\mathbf{I}') = u_j^*(\mathbf{I})$ for $j \leq M$ and $u_j(\mathbf{I}') = 0$ for all $j > M$. It is clear that $\{u_j(\mathbf{I}')\}_j$ is a valid solution of the LP on $\mathbf{I}'$, so we have

$$Q(\mathbf{I}', \tau) \geq \sum_j u_j(\mathbf{I}') = \sum_j u_j^*(\mathbf{I}) = Q(\mathbf{I}, \tau).$$

For the other direction, let $\{u_j^*(\mathbf{I}')\}_j$ be an optimal solution of the LP on $\mathbf{I}'$. We cut it down to a solution $\{u_j(\mathbf{I})\}_j$ of the LP on $\mathbf{I}$, by setting $u_j(\mathbf{I}) = u_j^*(\mathbf{I}')$ for $j \leq M$ while ignoring all $u_j^*(\mathbf{I}')$ for $j > M$. It is clear that $\{u_j(\mathbf{I})\}_j$ is a valid solution of the LP on $\mathbf{I}$, so we have

$$Q(\mathbf{I}, \tau) \geq \sum_j u_j(\mathbf{I}) \geq \sum_j u_j^*(\mathbf{I}') - \tau = Q(\mathbf{I}', \tau) - \tau,$$

where the second inequality follows from the observation that the constraint $\sum_{j \in C_{i^*}(\mathbf{I}')} u_j \leq \tau$ in the LP on $\mathbf{I}'$ implies that the sum of the ignored $u_j^*(\mathbf{I}')$'s is at most τ. □

Plugged into Theorem 5.1, this immediately yields the following corollary.

COROLLARY 5.4. *For any SJA query Q and any instance $\mathbf{I}$, R2T returns a $\tilde{Q}(\mathbf{I})$ such that, with probability at least $1 - \beta$,*

$$Q(\mathbf{I}) - 4\log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\frac{DS_Q(\mathbf{I})}{\varepsilon} \leq \tilde{Q}(\mathbf{I}) \leq Q(\mathbf{I}).$$

5.2 Truncation for SPJA Queries

A Negative Result. The correctness of the LP-based truncation mechanism relies on a key property of SJA queries, that removing t_P will always reduce $Q(\mathbf{I})$ by $S_Q(\mathbf{I}, t_P)$, which is the contribution of t_P to $Q(\mathbf{I})$. Unfortunately, the projection operator violates this property, as illustrated in the following example.

Example 5.5. Revisit the query Q in Example 1.1, where R_1 is the primary private relation and R_2 is a secondary relation. Consider the following instance $\mathbf{I}$: set $\mathbf{I}(R_1) = \{(a_1), (a_2)\}$, $\mathbf{I}(R_2) = \{(a_i, b_j) : i \in [2], j \in [m]\}$. Then, $S_Q(\mathbf{I}, (a_1)) = S_Q(\mathbf{I}, (a_2)) = m$, $Q(\mathbf{I}) = 2m$, and $DS_Q(\mathbf{I}) = m$.

Now, we add a projection operator, changing the query to

$$Q' := |\pi_{x_2}(R_1(x_1) \bowtie R_2(x_1, x_2))|.$$

Both (a_1) and (a_2) contribute m to $Q(\mathbf{I})$, but their contributions "overlap," thus removing either will not affect the query result (i.e., $DS_{Q'}(\mathbf{I}) = 0$). □

Intuitively, a projection reduces the query answer, hence its sensitivity, so it requires less noise. However, it makes achieving instance optimality harder because the optimality target, $DS_Q(\mathbf{I})$, may get a lot smaller, as illustrated in the preceding example. In particular, the second equality in (7) no longer holds (the first equality is the definition of $DS_Q(\mathbf{I})$), and $DS_Q(\mathbf{I})$ may be smaller than any $S_Q(\mathbf{I}, t_P)$. We formalize this intuition with the following negative result.

Theorem 5.6. *Let Q' be the query in Example 5.5. For any $GS_{Q'}$, there is a set of instances $\mathcal{I}$ with global sensitivity $GS_{Q'}$ such that, for any functions $M, f : \mathcal{I} \to \mathbb{R}$, if $\Pr[|M(\mathbf{I}) - Q'(\mathbf{I})| \leq f(\mathbf{I}) \cdot DS_{Q'}(\mathbf{I})] \geq 2/3$, then M is not ε-DP for any $\varepsilon < \frac{1}{2}\ln(2GS_{Q'})$.*

Proof. We build the set of instances $\mathcal{I}$ as follows. First, put the empty instance $\mathbf{I}_0$ into $\mathcal{I}$. Then, for any $m \in [GS_{Q'}]$, construct an $\mathbf{I}_m$ with $\mathbf{I}_m(R_1) = \{(a_1), (a_2)\}$, $\mathbf{I}_m(R_2) = \{(a_i, b_j) : i \in [2], j \in [m]\}$. Note that $Q'(\mathbf{I}_m) = m$, and $DS_{Q'}(\mathbf{I}_m) = 0$ since removing either (a_1) or (a_2) will not affect the query result. Finally, for each $\mathbf{I}_m$, remove (a_1) (and all referencing tuples) and add the resulting instance to $\mathcal{I}$. It can be verified that the global sensitivity of $\mathcal{I}$ is $GS_{Q'}$. Meanwhile, for any $m \in [GS_{Q'}]$, $\mathbf{I}_m$ and $\mathbf{I}_0$ are 2-hop neighbors, so if M is ε-DP, then

$$\Pr[M(\mathbf{I}_m) = y] \leq e^{2\varepsilon} \Pr[M(\mathbf{I}_0) = y],$$

for any y, by the *group privacy* property of DP [26].

The instance optimality guarantee implies that for every $m \in [GS_{Q'}]$,

$$\Pr[M(\mathbf{I}_m) = m] \geq 2/3.$$

Consider $\mathbf{I}_0$. On the one hand,

$$\Pr[M(\mathbf{I}_0) \neq 0] \leq 1/3. \tag{12}$$

On the other hand,

$$\begin{aligned}\Pr[M(\mathbf{I}_0) \neq 0] &\geq \Pr[M(\mathbf{I}_0) = 1] + \cdots + \Pr[M(\mathbf{I}_0) = GS_{Q'}] \\ &\geq \sum_{m=1}^{GS_{Q'}} e^{-2\varepsilon} \Pr[M(\mathbf{I}_m) = m] \\ &\geq \sum_{m=1}^{GS_{Q'}} e^{-2\varepsilon} \cdot 2/3 = \frac{2GS_{Q'}}{3e^{2\varepsilon}},\end{aligned}$$

which contradicts (12) when $\varepsilon < \frac{1}{2}\ln(2GS_{Q'})$. □

Indirect Sensitivity. Recall the definition of $S_Q(\mathbf{I}, t_P)$ as in (5). However, for an SPJA query, we have $Q(\mathbf{I}) = \sum_{q \in \pi_{\mathbf{y}} J(\mathbf{I})} \psi(q)$ instead of $Q(\mathbf{I}) = \sum_{q \in J(\mathbf{I})} \psi(q)$, thus (7) no longer holds. This means that while $S_Q(\mathbf{I}, t_P)$ is still the contribution of t_P to $Q(\mathbf{I})$, it is "indirect": the overlapping contributions should be counted only once due to the projection operator removing duplicates.

We now define the *indirect sensitivity* for an instance $\mathbf{I}$:

$$IS_Q(\mathbf{I}) = \max_{t_P \in \mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P).$$

It should be clear that $IS_Q(\mathbf{I}) \geq DS_Q(\mathbf{I})$ due to the overlapping; in the extreme case shown in Example 5.5, we have $IS_Q(\mathbf{I}) = m$ but $DS_Q(\mathbf{I}) = 0$. In the following, we give a truncation method for SPJA queries with $\tau^*(\mathbf{I}) = IS_Q(\mathbf{I})$. When plugged into R2T, this yields a DP mechanism with error $O\left(\log(GS_Q) \log\log(GS_Q) IS_Q(\mathbf{I})/\varepsilon\right)$. This is not instance-optimal, which is unachievable by

Theorem 5.6 anyway. Note that for SJA queries, we have $\mathbf{y} = var(J)$, and $DS_Q(\mathbf{I}) = IS_Q(\mathbf{I})$ in this case.

Truncation Mechanism. We modify the LP-based truncation mechanism from Section 5.1 to handle SPJA queries. Let $L = |\pi_{\mathbf{y}} J(\mathbf{I})|$ and $p_k(\mathbf{I})$ be the k-th result in $\pi_{\mathbf{y}} J(\mathbf{I})$. To formalize the relationship of the query results before and after the projection, we use $E_k(\mathbf{I})$ to denote (the indices of) the join results corresponding to the projected result $p_k(\mathbf{I})$—that is,

$$E_k(\mathbf{I}) := \{j : p_k = \pi_{\mathbf{y}} q_j(\mathbf{I})\},$$

whereas $C_i(\mathbf{I})$ is still defined as in (2).

Now, we define a new LP. For each $k \in [L]$, we introduce a new variable $v_k \in [0, \psi(p_k(\mathbf{I}))]$, which represents the weight assigned to the projected result $p_k(\mathbf{I})$. For each $j \in [M]$, we still use a variable $u_j(\mathbf{I}) \in [0, \psi(q_j(\mathbf{I}))]$ to represent the weight assigned to $q_j(\mathbf{I})$. We keep the same truncation constraints on the u_j's, while adding the constraint that the weight of a projected result should not exceed the total weights of all its corresponding join results. Then, we try to maximize the projected results. More precisely, the new LP is

$$\begin{aligned}
\text{maximize} \quad & Q(\mathbf{I}, \tau) = \sum_{k \in [L]} v_k \\
\text{subject to} \quad & v_k \leq \sum_{j \in E_k(\mathbf{I})} u_j, \qquad k \in [L], \\
& \sum_{j \in C_i(\mathbf{I})} u_j \leq \tau, \qquad i \in [N], \\
& 0 \leq u_j \leq \psi(q_j(\mathbf{I})), \ j \in [M], \\
& 0 \leq v_k \leq \psi(p_k(\mathbf{I})), k \in [L].
\end{aligned}$$

We can show that this modified LP yields a valid truncation method for SPJA queries.

LEMMA 5.7. *For SPJA queries, the $Q(\mathbf{I}, \tau)$ defined earlier satisfies the three properties required by R2T with $\tau^*(\mathbf{I}) = IS_Q(\mathbf{I})$.*

PROOF. First, same as SJA queries, property (2) holds due to the constraint $v_l \leq \psi(p_l(\mathbf{I}))$. For property (3), we have $S_Q(\mathbf{I}, t_i) = \sum_{j \in C_i(\mathbf{I})} \psi(q_j(\mathbf{I}))$. Then, with the same argument as in the proof of Lemma 5.3, we can show that the property holds with $\tau^*(\mathbf{I}) = IS_Q(\mathbf{I})$. Finally, consider property (1). For any $\mathbf{I} \sim \mathbf{I}', \mathbf{I} \subseteq \mathbf{I}'$, it is easy to see that $J(\mathbf{I}) \subseteq J(\mathbf{I}')$ and all different projection results are in C_{i^*} for some $i^* \in N$. Then, the same line of reasoning as in the proof of Lemma 5.3 proves property (1). □

Plugged into Theorem 5.1, we have the following corollary.

COROLLARY 5.8. *For any SPJA query Q and any instance $\mathbf{I}$, R2T returns a $\tilde{Q}(\mathbf{I})$ such that, with probability at least $1 - \beta$,*

$$Q(\mathbf{I}) - 4 \log(GS_Q) \ln\left(\frac{\log(GS_Q)}{\beta}\right) \frac{IS_Q(\mathbf{I})}{\varepsilon} \leq \tilde{Q}(\mathbf{I}) \leq Q(\mathbf{I}).$$

6 OPT^2: Optimal Optimality Ratio without Assumptions

In this section, we propose another DP mechanism for answering SJA queries that improves upon R2T from the following two aspects. First, R2T relies on assuming a finite GS_Q. This is undesirable both theoretically and practically. Theoretically, this assumption restricts the space of allowable database instances. Practically, setting an appropriate GS_Q is not easy, as one can never foresee how much data an individual may possess (via FKs) in any database instance. Note that it is wrong to set GS_Q to be the largest number of tuples belonging to an individual in the given instance $\mathbf{I}$. It must be set in such a way that (4) holds for all instances on which the mechanism is *ever* going to be applied.

The mechanism described in this section not only removes the assumption on GS_Q, thus allowing all database instances (FK constraints must still be satisfied), but also improves the optimality ratio from $O(\log(GS_Q)\log\log(GS_Q))$ to $O\left(\log\log(DS_Q(\mathbf{I}))\right)$. This doubly logarithmic optimality ratio has recently been shown to be optimal even for self-join-free queries—that is, the sum estimation [22]. We thus call this new mechanism OPT^2—namely, it achieves down-neighborhood optimality with an optimal optimality ratio.

In the following, we first present a mechanism achieving the preceding two goals, but it takes exponential time. Then, we show how to reduce the running time to polynomial by using LPs. Finally, we show how the mechanism can be applied on SPJA queries as well.

6.1 An Exponential Time Algorithm for SJA Queries

In some sense, R2T bypasses the τ selection step and returns a privatized query answer directly. In OPT^2, we first select a good τ by finding a measurement $G(\mathbf{I},\tau)$ on any given τ. More importantly, we will design $G(\mathbf{I},\tau)$ with a small global sensitivity so that we can use the privatized values of $G(\mathbf{I},\tau)$ to do the τ selection with the SVT technique.

For any $\tau \in \mathbb{N}$, we define $F(\mathbf{I},\tau)$ as the maximum size of the primary private relation instance over any $\mathbf{I}'' \subseteq \mathbf{I}$ whose downward local sensitivity is bounded by τ—that is,

$$F(\mathbf{I},\tau) = \max_{\mathbf{I}''\subseteq \mathbf{I},\, DS_Q(\mathbf{I}'')\le\tau} |\mathbf{I}''(R_P)|. \tag{13}$$

For now, we use a brute-force method to compute $F(\mathbf{I},\tau)$ by enumerating all $\mathbf{I}'' \subseteq \mathbf{I}$, thus taking exponential time.

Next, define

$$G(\mathbf{I},\tau) = F(\mathbf{I},\tau) - N.$$

The following observation is immediate.

Lemma 6.1. *For any* $\mathbf{I}$ *and any* τ*,* $G(\mathbf{I},\tau) \le 0$*. If* $\tau \ge DS_Q(\mathbf{I})$*, then* $G(\mathbf{I},\tau) = 0$*.*

Therefore, we can decide whether τ is a good truncation threshold by looking at $G(\mathbf{I},\tau)$: if $G(\mathbf{I},\tau)$ is close to 0, using τ to the truncation will only result in a few tuples being truncated.

We next show that $G(\cdot,\tau)$ has small global sensitivity for any τ.

Lemma 6.2. *For any* $\tau \in \mathbb{N}$*,* $G(\cdot,\tau)$ *has global sensitivity* 1*.*

Proof. Given τ, for any $\mathbf{I} \sim \mathbf{I}'$, assume $\mathbf{I}'(R_P) = \mathbf{I}(R_P) \cup \{t_P\}$ without loss of generality. On the one hand, it is trivial to see

$$F(\mathbf{I}',\tau) \ge F(\mathbf{I},\tau), \tag{14}$$

since for any $\mathbf{I}'' \subseteq \mathbf{I}$, we also have $\mathbf{I}'' \subseteq \mathbf{I}'$. On the other hand, let $\mathbf{I}^{*\prime} = \arg\max_{\mathbf{I}''\subseteq\mathbf{I}',\, DS_Q(\mathbf{I}'')\le\tau} |\mathbf{I}''|$. Then, we can construct a $\mathbf{I}^*$ from $\mathbf{I}^{*\prime}$ by deleting all tuples referencing t_P. Then, $\mathbf{I}^* \subseteq \mathbf{I}$, $DS_Q(\mathbf{I}^*) \le DS_Q(\mathbf{I}^{*\prime}) \le \tau$ and $|\mathbf{I}^*(R_P)| \ge |\mathbf{I}^{*\prime}(R_P)| - 1$, which means

$$F(\mathbf{I},\tau) \ge F(\mathbf{I}',\tau) - 1. \tag{15}$$

Finally, combining (14), (15), and $N' = N + 1$, we can get

$$G(\mathbf{I},\tau) - 1 \le G(\mathbf{I}',\tau) \le G(\mathbf{I},\tau). \qquad \square$$

ALGORITHM 2: ExpOPT^2

Input: $\mathbf{I}, \varepsilon, \beta, Q$

$\tilde{\ell} \leftarrow \text{SVT}\,(-9\ln(4/\beta)/\varepsilon, 2\varepsilon/3, G(\mathbf{I}, 2), G(\mathbf{I}, 4), G(\mathbf{I}, 8), \ldots);$

$\tilde{\tau} \leftarrow 2^{\tilde{\ell}};$

$\tilde{Q}(\mathbf{I}) \leftarrow Q(\mathbf{I}, \tilde{\tau}) + Lap\left(\frac{3\tilde{\tau}}{\varepsilon}\right);$

return $\tilde{Q}(\mathbf{I})$;

We can thus privately select a $\tilde{\tau}$ such that $G(\mathbf{I}, \tilde{\tau})$ is smaller but very close to 0. The idea is to run SVT over the queries $G(\mathbf{I}, 2), G(\mathbf{I}, 4), G(\mathbf{I}, 8), \ldots$ with the threshold $T = -9\ln(4/\beta)/\varepsilon$. After selecting a privatized threshold $\tilde{\tau}$, we truncate with the LP for SJA queries introduced in Section 5.1 and finally add $Lap(\tilde{\tau}/\varepsilon)$. The details of the algorithm, called *ExpOPT*^{2}, are shown in Algorithm 2.

Example 6.3. Here, we give a step-by-step example to show how ExpOPT^2 works. Follow Example 5.2 where $N = 8103$ and $Q(\mathbf{I}) = 9992$. We also set $\beta = 0.1$ and $\varepsilon = 1$.

We run ExpOPT^2 with $\tau = 2, 4, 8, \ldots$ To calculate $F(\mathbf{I}, \tau)$, we consider each clique/star separately. For each k-clique instance $\mathbf{I}_{C_k}$, we can get $F(\mathbf{I}_{C_k}, \tau) = \min(\tau + 1, k)$. For each k-star instance $\mathbf{I}_{S_k}$, we have $F(\mathbf{I}_{S_k}, \tau) = \min(\tau + 1, k + 1)$. Thus, we can compute

$$F(\mathbf{I}, 2) = 3 \times 1000 + 3 \times 1000 + 3 \times 100 + 3 \times 10 + 3 \times 1 = 6333,$$
$$F(\mathbf{I}, 4) = 3 \times 1000 + 4 \times 1000 + 5 \times 100 + 5 \times 10 + 5 \times 1 = 7555,$$
$$F(\mathbf{I}, 8) = 3 \times 1000 + 4 \times 1000 + 9 \times 100 + 9 \times 10 + 9 \times 1 = 7999,$$
$$F(\mathbf{I}, 16) = 3 \times 1000 + 4 \times 1000 + 9 \times 100 + 17 \times 10 + 17 \times 1 = 8087.$$

Besides, we have $F(\mathbf{I}, \tau) = 8103$ for $\tau \geq 32$.

We then compute $G(\mathbf{I}, \tau) = F(\mathbf{I}, \tau) - N$ for all τ's and run the SVT. Similar to Example 5.2, we assume $Lap(1)$ returns $\{-1, 1\}$ by turns. In SVT,

$$\tilde{T} = -9\ln(4/\beta)/\varepsilon + Lap(3/\varepsilon) = -33.2 + (-1) \cdot 3 = -36.2,$$

and

$$\tilde{Q}_1(\mathbf{I}) = F(\mathbf{I}, 2) - N + Lap(6) = 6333 - 8103 + 1 \cdot 6 = -1764,$$
$$\tilde{Q}_2(\mathbf{I}) = F(\mathbf{I}, 4) - N + Lap(6) = 7555 - 8103 + (-1) \cdot 6 = -554,$$
$$\tilde{Q}_3(\mathbf{I}) = F(\mathbf{I}, 8) - N + Lap(6) = 7999 - 8103 + 1 \cdot 6 = -98,$$
$$\tilde{Q}_4(\mathbf{I}) = F(\mathbf{I}, 16) - N + Lap(6) = 8087 - 8103 + (-1) \cdot 6 = -22.$$

Therefore, we will select $\tilde{\tau} = 16$. Finally, we compute

$$\tilde{Q}(\mathbf{I}) = Q(\mathbf{I}, \tilde{\tau}) + Lap(\frac{3\tilde{\tau}}{\varepsilon}) = 9976 + 1 \cdot 48 = 10024. \qquad \square$$

The privacy analysis of ExpOPT^2 is straightforward. By Lemma 6.2 and Lemma 3.5, each $G(\mathbf{I}, \cdot)$ has the sensitivity bounded by 1 and thus the SVT is $(2\varepsilon/3)$-DP. Besides, by Lemma 5.3 and Lemma 3.4, $Q(\mathbf{I}, \tilde{\tau})$ has the sensitivity bounded by $\tilde{\tau}$, so $\tilde{Q}(\mathbf{I})$ preserves $(\varepsilon/3)$-DP. Then, we obtain that ExpOPT^2 preserves ε-DP by the basic composition theorem of DP [26].

Utility Analysis. The intuition why ExpOPT^2 offers good utility is as follows. When $\tau \geq DS_Q(\mathbf{I})$, we always have $F(\mathbf{I}, \tau) = N$. Then according to Lemma 3.5, we know that with constant probability, only a few (i.e., $O\left(\log\log(DS_Q(\mathbf{I}))\right)$) tuples t_P's satisfy $S_Q(\mathbf{I}, t_P) > \tilde{\tau}$. Therefore, truncating with $\tilde{\tau}$ will only lead to a bias of $O\left(\log\log(DS_Q(\mathbf{I}))DS_Q(\mathbf{I})\right)$. Besides, we have $\tilde{\tau} = O(DS_Q(\mathbf{I}))$ so that the noise is bounded by $O(DS_Q(\mathbf{I}))$. Overall, we can bound the error by $O\left(\log\log(DS_Q(\mathbf{I}))\cdot DS_Q(\mathbf{I})\right)$. More precisely, we show that its over-estimation is $O(DS_Q(\mathbf{I}))$ while the under-estimation is $O\left(\log\log(DS_Q(\mathbf{I}))\cdot DS_Q(\mathbf{I})\right)$.

Theorem 6.4. *On any instance* $\mathbf{I}$, *ExpOPT*^{2} *returns a* $\tilde{Q}(\mathbf{I})$ *such that with probability at least* $1-\beta$,

$$Q(\mathbf{I}) - \frac{24DS_Q(\mathbf{I})}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right) \le \tilde{Q}(\mathbf{I}) \le Q(\mathbf{I}) + \frac{6DS_Q(\mathbf{I})}{\varepsilon}\ln\left(\frac{2}{\beta}\right).$$

Proof. First, by Lemma 3.5 and 6.1, with probability at least $1-\frac{\beta}{2}$, we have

$$\tilde{\tau} \le 2DS_Q(\mathbf{I}), \tag{16}$$

and

$$G(\mathbf{I},\tilde{\tau}) \ge -\frac{9}{\varepsilon}\ln\left(\frac{4}{\beta}\right) - \frac{9}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right),$$

which further means

$$N - F(\mathbf{I},\tilde{\tau}) \le \frac{18}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right). \tag{17}$$

Recall

$$F(\mathbf{I},\tilde{\tau}) = \max_{\mathbf{I}''\subseteq\mathbf{I},\, DS_Q(\mathbf{I}'')\le\tilde{\tau}} |\mathbf{I}''(R_P)|,$$

and we denote

$$\mathbf{I}^* = \underset{\mathbf{I}''\subseteq\mathbf{I},\, DS_Q(\mathbf{I}'')\le\tilde{\tau}}{\arg\max} |\mathbf{I}''(R_P)|.$$

Then, by definition of $\mathbf{I}^*$ and (17), we have

$$|\mathbf{I}(R_P)| - |\mathbf{I}^*(R_P)| \le \frac{18}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right),$$

and for any $R\in\mathbf{R}-\{R_P\}$, $\mathbf{I}(R)=\mathbf{I}^*(R)$. Further recall that by the definition of $DS_Q(\mathbf{I})$, we can get

$$Q(\mathbf{I}) - Q(\mathbf{I}^*) \le \frac{18DS_Q(\mathbf{I})}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right). \tag{18}$$

Recall $Q(\mathbf{I},\tilde{\tau})$ is the optimal solution of the LP defined in Section 5.1. We now show that we can construct a valid solution of the LP based on $\mathbf{I}^*$. Since $\mathbf{I}^*\subseteq\mathbf{I}$, $\mathbf{I}^*(R_P)\subseteq\mathbf{I}(R_P)$ and $J(\mathbf{I}^*)\subseteq J(\mathbf{I})$. Then, we set $u_j=\psi(q_j(\mathbf{I}))$ if $q_j\in J(\mathbf{I}^*)$ and $u_j=0$ otherwise. We can thus ensure $u_j\in[0,\psi(q_j(\mathbf{I}))]$ for any $j\in[M]$. Moreover, $DS_Q(\mathbf{I}^*)\le\tilde{\tau}$ implies $\sum_{j\in C_i(\mathbf{I})}u_j\le\tilde{\tau}$ for any $i\in[N]$. Therefore, $\{u_j\}_j$ is a valid solution for the LP of $Q(\mathbf{I},\tilde{\tau})$ and its objective value is equal to $Q(\mathbf{I}^*)$. Since the LP maximizes the objective function, we have

$$Q(\mathbf{I}^*) \le Q(\mathbf{I},\tilde{\tau}) \le Q(\mathbf{I}), \tag{19}$$

where the second inequality is by Lemma 5.3. Combining (18) and (19), we have

$$Q(\mathbf{I}) - Q(\mathbf{I},\tilde{\tau}) \le \frac{18DS_Q(\mathbf{I})}{\varepsilon}\ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right).$$

Finally, we complete the proof by using the tail bound of the Laplace distribution to show, with probability at least $1-\frac{\beta}{2}$,

$$|\tilde{Q}(\mathbf{I}) - Q(\mathbf{I},\tilde{\tau})| \le \frac{3\tilde{\tau}}{\varepsilon}\ln\left(\frac{2}{\beta}\right) \le \frac{6DS_Q(\mathbf{I})}{\varepsilon}\ln\left(\frac{2}{\beta}\right),$$

where the second equality is by (16). □

6.2 A Polynomial Time Algorithm for SJA Queries

The computational bottleneck in ExpOPT[2] is the brute-force algorithm for computing $F(\mathbf{I}, \tau)$. Unfortunately, there is little hope to do better than this, because even for the edge-counting query under node-DP, which is a special SJA query, computing $F(\mathbf{I}, \tau)$ already requires us to solve the *maximum degree-bounded induced subgraph* problem, which is an NP-hard problem and even hard to approximate [37].

To get around this difficulty, we will replace $F(\mathbf{I}, \tau)$ by a proxy $\hat{F}(\mathbf{I}, \tau)$ that is efficiently computable. Our key observation is that $\hat{F}(\mathbf{I}, \tau)$ does not need to be an approximation of $F(\mathbf{I}, \tau)$ in the traditional sense of approximation algorithms. In fact, we do not even need $\hat{F}(\mathbf{I}, \tau)$ to correspond to a valid $\mathbf{I}''$ as in the definition of $F(\mathbf{I}, \tau)$ in (13) or take integer values. Instead, we only need $\hat{F}(\mathbf{I}, \tau)$, hence $\hat{G}(\mathbf{I}, \tau)$, to satisfy Lemma 6.1 and 6.2.

We will define $\hat{F}(\mathbf{I}, \tau)$ using LP relaxation. First, we write $F(\mathbf{I}, \tau)$ as an ILP. To represent an $\mathbf{I}'' \subseteq \mathbf{I}$, for each tuple t_i in the primary private relation, we introduce a variable y_i to indicate whether it is included in $\mathbf{I}''(R_P)$. Similarly, we introduce a variable z_j for each q_j to indicate whether it is included in $J(\mathbf{I}'')$. Recall that $D_j(\mathbf{I})$ denotes the indices of the set of tuples that $q_j(\mathbf{I})$ references. For each z_j, let $z_j \geq \sum_{i \in D_j(\mathbf{I})} y_i - |D_j(\mathbf{I})| + 1$. This is to ensure that q_j appears in $J(\mathbf{I}'')$ when all t_i's for $i \in D_j(\mathbf{I})$ are included in $R_P(\mathbf{I}'')$. To enforce $DS_Q(\mathbf{I}'') \leq \tau$, we add the linear constraint $\sum_{j \in C_i(\mathbf{I})} (z_j \cdot \psi(q_j(\mathbf{I}))) \leq \tau$ for all $i \in [N]$. Above all, the ILP is written as

$$
\begin{aligned}
\text{maximize} \quad & F(\mathbf{I}, \tau) = \sum_{i \in [N]} y_i \\
\text{subject to} \quad & z_j \geq \sum_{i \in D_j(\mathbf{I})} y_i - |D_j(\mathbf{I})| + 1, && j \in [M], \\
& \sum_{j \in C_i(\mathbf{I})} (z_j \cdot \psi(q_j(\mathbf{I}))) \leq \tau, && i \in [N], \\
& y_i \in \{0, 1\}, && i \in [N], \\
& z_j \in \{0, 1\}, && j \in [M].
\end{aligned}
$$

Next, we relax it into an LP:

$$
\begin{aligned}
\text{maximize} \quad & \hat{F}(\mathbf{I}, \tau) = \sum_{i \in [N]} y_i, \\
\text{subject to} \quad & z_j \geq \sum_{i \in D_j(\mathbf{I})} y_i - |D_j(\mathbf{I})| + 1, && j \in [M], \\
& \sum_{j \in C_i(\mathbf{I})} (z_j \cdot \psi(q_j(\mathbf{I}))) \leq \tau, && i \in [N], \\
& y_i \in [0, 1], && i \in [N], \\
& z_j \in [0, 1], && j \in [M].
\end{aligned}
$$

Finally, set

$$\hat{G}(\mathbf{I}, \tau) = \hat{F}(\mathbf{I}, \tau) - N.$$

Now, we show that Lemma 6.1 and 6.2 still hold for $\hat{G}(\mathbf{I}, \tau)$.

LEMMA 6.5. *For any* $\mathbf{I}$ *and any* τ, $\hat{G}(\mathbf{I}, \tau) \leq 0$ *and if* $\tau \geq DS_Q(\mathbf{I})$, *then* $\hat{G}(\mathbf{I}, \tau) = 0$.

The proof of Lemma 6.5 is still trivial.

LEMMA 6.6. *For any* $\tau \in \mathbb{N}$, $\hat{G}(\cdot, \tau)$ *has global sensitivity* 1.

Proof. Similar to the proof of Lemma 6.2, for any $\mathbf{I} \sim \mathbf{I}'$, assume $\mathbf{I}'(R_P) = \mathbf{I}(R_P) \cup \{t_P\}$ and it suffices to show

$$\hat{F}(\mathbf{I}', \tau) \geq \hat{F}(\mathbf{I}, \tau) \geq \hat{F}(\mathbf{I}', \tau) - 1.$$

Let $\{y_i^*(\mathbf{I})\}_i$, $\{z_j^*(\mathbf{I})\}_j$ be the optimal solution of $\hat{F}(\mathbf{I}, \tau)$, and $\{y_i^*(\mathbf{I}')\}_i$, $\{z_j^*(\mathbf{I}')\}_j$ the optimal solution of $\hat{F}(\mathbf{I}', \tau)$. Similar to the proof of Lemma 5.3, $J(\mathbf{I}) \subseteq J(\mathbf{I}')$ and we assume that the extra join results in $J(\mathbf{I}')$ are put at the end.

For the first inequality, we can extend $\{y_i^*(\mathbf{I})\}_i$, $\{z_j^*(\mathbf{I})\}_j$ to a valid solution $\{y_i(\mathbf{I}')\}_i$, $\{z_j(\mathbf{I}')\}_j$ of $\mathbf{I}'$ by setting $y_i(\mathbf{I}') = 0$ for t_P and $z_j(\mathbf{I}') = 0$ for all $j > M$. For the second inequality, we can cut $\{y_i^*(\mathbf{I}')\}_i$, $\{z_j^*(\mathbf{I}')\}_j$ down to a valid solution $\{y_i(\mathbf{I})\}_i$, $\{z_j(\mathbf{I})\}_j$ of $\mathbf{I}$ by ignoring $y_i^*(\mathbf{I}')$ for t_P and $z_j^*(\mathbf{I}')$ for $j > M$. Then, $\sum_{i\in[N]} y_i(\mathbf{I}) \geq \sum_{i\in[N']} y_i^*(\mathbf{I}') - 1$. □

Then, the algorithm OPT^2 is the same as ExpOPT^2 except that $G(\mathbf{I}, \tau)$ is replaced with $\hat{G}(\mathbf{I}, \tau)$.

Example 6.7. We give a step-by-step example to show how OPT^2 works. Consider the same problem of edge counting under node-DP and the same instance $\mathbf{I}$ as Example 5.2. We have $N = 8103$ and $Q(\mathbf{I}) = 9992$. We again set $\beta = 0.1$ and $\varepsilon = 1$.

We run OPT^2 with $\tau = 2, 4, 8\ldots$ For each τ, we assign a weight $y_i \in [0, 1]$ to each node, and a weight $z_j \in [0, 1]$ to each edge that satisfies the predicate `ID1 < ID2`. To calculate $\hat{F}(\mathbf{I}, \tau)$, we again can consider the LP on each clique/star separately. For a triangle, the optimal LP solution assigns $z_j = \tau/2$ for each edge and $y_i = 1/2 + \tau/4$ for each node when $\tau < 2$ and $z_j = y_i = 1$ when $\tau \geq 2$. For each 4-clique, $z_j = \tau/3$ for each edge and $y_i = 1/2 + \tau/6$ for each node when $\tau < 3$ and $z_j = y_i = 1$ when $\tau \geq 3$. For each k-star, the LP optimal solution is $k + \min(\tau/k, 1)$. Thus, the optimal LP solutions are

$$\begin{aligned}
\hat{F}(\mathbf{I}, 2) &= 1 \times 3000 + \frac{5}{6} \times 4000 + 8\frac{1}{4} \times 100 + 16\frac{1}{8} \times 10 + 32\frac{1}{16} \times 1 \approx 7351.646,\\
\hat{F}(\mathbf{I}, 4) &= 1 \times 3000 + 1 \times 4000 + 8\frac{1}{2} \times 100 + 16\frac{1}{4} \times 10 + 32\frac{1}{8} \times 1 = 8044.625,\\
\hat{F}(\mathbf{I}, 8) &= 1 \times 3000 + 1 \times 4000 + 9 \times 100 + 16\frac{1}{2} \times 10 + 32\frac{1}{4} \times 1 = 8097.25,\\
\hat{F}(\mathbf{I}, 16) &= 1 \times 3000 + 1 \times 4000 + 9 \times 100 + 17 \times 10 + 32\frac{1}{2} \times 1 = 8102.5.
\end{aligned}$$

In addition, we have $\hat{F}(\mathbf{I}, \tau) = 8103$ for $\tau \geq 32$. We can see that $\hat{F}(\mathbf{I}, \tau) \geq F(\mathbf{I}, \tau)$ for any τ due to the relaxation of the ILP.

We then compute $G(\mathbf{I}, \tau) = F(\mathbf{I}, \tau) - N$ for all τ's and run the SVT. Assume $Lap(1)$ returns $\{-1, 1\}$ by turns. We have

$$\tilde{T} = -9\ln(4/\beta)/\varepsilon + Lap(3/\varepsilon) = -33.2 + (-1) \cdot 3 = -36.2,$$

and

$$\begin{aligned}
\tilde{Q}_1(\mathbf{I}) =& \hat{F}(\mathbf{I}, 2) - N + Lap(6) = 7351.646 - 8103 + 1 \cdot 6 = -745.354,\\
\tilde{Q}_2(\mathbf{I}) =& \hat{F}(\mathbf{I}, 4) - N + Lap(6) = 8044.625 - 8103 + (-1) \cdot 6 = -64.375,\\
\tilde{Q}_3(\mathbf{I}) =& \hat{F}(\mathbf{I}, 8) - N + Lap(6) = 8097.25 - 8103 + 1 \cdot 6 = 0.25,\\
\tilde{Q}_4(\mathbf{I}) =& \hat{F}(\mathbf{I}, 16) - N + Lap(6) = 8102.5 - 8103 + (-1) \cdot 6 = -6.5.
\end{aligned}$$

After selecting the privatized threshold, which is $\tilde{\tau} = 8$ here, we finally compute

$$\tilde{Q}(\mathbf{I}) = Q(\mathbf{I}, \tilde{\tau}) + Lap(\frac{3\tilde{\tau}}{\varepsilon}) = 9888 + 1 \cdot 24 = 9912.$$ □

Utility Analysis. Finally, we can show the utility guarantee still holds.

THEOREM 6.8. *On any instance* $\mathbf{I}$, *OPT*^{2} *returns a* $\tilde{Q}(\mathbf{I})$ *such that with probability at least* $1-\beta$,

$$|\tilde{Q}(\mathbf{I}) - Q(\mathbf{I})| \leq \frac{24DS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right).$$

PROOF. First, similar to the proof of Theorem 6.4, by Lemma 3.5 and 6.5, with probability at least $1-\frac{\beta}{2}$,

$$\tilde{\tau} \leq 2DS_Q(\mathbf{I}), \tag{20}$$

and

$$N - \hat{F}(\mathbf{I}, \tilde{\tau}) \leq \frac{18}{\varepsilon} \ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right). \tag{21}$$

Let $\{y_i^*(\mathbf{I})\}_i$, $\{z_j^*(\mathbf{I})\}_j$ be the corresponding solutions of $\hat{F}(\mathbf{I}, \tilde{\tau})$. First, by (21), we have

$$N - \sum_i y_i^*(\mathbf{I}) \leq \frac{18}{\varepsilon} \ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right). \tag{22}$$

Let $V^*(\mathbf{I}) = \sum_j (z_j^*(\mathbf{I}) \cdot \psi(q_j(\mathbf{I})))$. We next show

$$Q(\mathbf{I}) - V^*(\mathbf{I}) \leq \frac{18DS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right). \tag{23}$$

We prove it by increasing the values of $y_i^*(\mathbf{I})$'s, $z_j^*(\mathbf{I})$'s, and $V^*(\mathbf{I})$ iteratively. For convenience, we use superscript to show the iteration and regard the original ones as the values at the iteration 0—that is, let $y_i^{*(0)}(\mathbf{I}) = y_i^*(\mathbf{I}), z_j^{*(0)}(\mathbf{I}) = z_j^*(\mathbf{I})$ for any $i \in [N], j \in [M]$ and $V^{*(0)}(\mathbf{I}) = V^*(\mathbf{I})$. At iteration ℓ, we increase $y_i^{*(\ell-1)}(\mathbf{I})$ to 1 for $i = \ell$. Meanwhile, we update $z_j^{*(\ell)}(\mathbf{I}) = \sum_{i \in D_j(\mathbf{I})} y_i^{*(\ell)}(\mathbf{I}) - |D_j(\mathbf{I})| + 1$ for each $j \in [M]$ and $V^{*(\ell)}(\mathbf{I}) = \sum_j (z_j^{*(\ell)}(\mathbf{I}) \cdot \psi(q_j(\mathbf{I})))$ correspondingly. Recalling the definition of $DS_Q(\mathbf{I})$, we thus have

$$V^{*(\ell)}(\mathbf{I}) - V^{*(\ell-1)}(\mathbf{I}) \leq DS_Q(\mathbf{I}) \cdot (1 - y_i^{*(\ell-1)}(\mathbf{I})). \tag{24}$$

After all iterations, since $y_i^{*(N)}(\mathbf{I}) = 1$ for any $i \in [N]$, we have $z_j^{*(N)} = 1$ for any $j \in [M]$, which further means

$$V^{*(N)}(\mathbf{I}) = Q(\mathbf{I}). \tag{25}$$

Finally, combining (22), (24), and (25), we can derive (23).

Now, let us compare $V^*(\mathbf{I})$ and $Q(\mathbf{I}, \tilde{\tau})$. Here, we can construct a valid solution of LP corresponding to $Q(\mathbf{I}, \tilde{\tau})$ based on $\{y_i^*(\mathbf{I})\}_i$, $\{z_j^*(\mathbf{I})\}_j$. For each j, let $u_j(\mathbf{I}) = z_j^*(\mathbf{I}) \cdot \psi(q_j(\mathbf{I}))$. Since $z_j^*(\mathbf{I}) \in [0, 1]$, we can ensure $u_j(\mathbf{I}) \in [0, \psi(q_j(\mathbf{I}))]$ for any $j \in [M]$. The constraints $\sum_{j \in C_i(\mathbf{I})} (z_j^*(\mathbf{I}) \cdot \psi(q_j(\mathbf{I}))) \leq \tilde{\tau}$ for all $i \in [N]$ implies $\sum_{j \in C_i(\mathbf{I})} u_j(\mathbf{I}) \leq \tilde{\tau}$. Therefore, $\{u_j(\mathbf{I})\}_j$ is a valid solution of the LP of $Q(\mathbf{I}, \tilde{\tau})$ and the corresponding objective value is $V^*(\mathbf{I})$. Above all,

$$V^*(\mathbf{I}) \leq Q(\mathbf{I}, \tilde{\tau}) \leq Q(\mathbf{I}). \tag{26}$$

Combining (23) and (26), we have

$$|Q(\mathbf{I}) - Q(\mathbf{I}, \tilde{\tau})| \leq \frac{18DS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2DS_Q(\mathbf{I}))}{\beta}\right).$$

With the same procedure as the proof of Theorem 6.4, we can derive the target error bound for $\tilde{Q}(\mathbf{I})$. □

6.3 Handling SPJA Queries

While ExpOPT^2 and OPT^2 achieve instance optimality in answering arbitrary SJA queries, unfortunately it cannot be directly applied to SPJA queries since they breach privacy, as illustrated in the following example.

Example 6.9. Recall the SPJA query Q' in Example 5.5. Given m, n, consider the instance $\mathbf{I}$ where $\mathbf{I}(R_1) = \{(a_i) : i \in [n]\}$, $\mathbf{I}(R_2) = \{(a_i, b_j) : i \in [n], j \in [(i-1)m+1, im]\}$, and the instance $\mathbf{I}'$ where $\mathbf{I}(R_1) = \{(a_i) : i \in [n+1]\}$, $\mathbf{I}(R_2) = \{(a_i, b_j) : i \in [n], j \in [(i-1)m+1, im]\} \cup \{(a_{n+1}, b_j) : j \in [nm]\}$. It is trivial that $\mathbf{I} \sim \mathbf{I}', \mathbf{I} \subseteq \mathbf{I}'$.

We can compute $DS_{Q'}(\mathbf{I}') = 0$ and $DS_{Q'}(\mathbf{I}) = m$. When we apply ExpOPT^2, for any $\tau < m$, we get $F(\mathbf{I}', \tau) = n+1$ and $F(\mathbf{I}, \tau) = 0$. Note that both m and n can be arbitrarily large, and therefore we cannot bound the sensitivity of $F(\cdot, \tau)$ for any τ. Thus, Lemma 6.2 no longer holds, and SVT does not satisfy DP. □

This is because the downward local sensitivity may decrease a lot when some users are inserted and Lemma 6.2 no longer holds. For OPT^2, the problem is more challenging. Recall the idea of OPT^2 is to formulate $F(\mathbf{I}, \tau)$ as an ILP and relax that to an LP. However, for SPJA queries, we even do not know how to formulate $F(\mathbf{I}, \tau)$ as an ILP.

To address this issue, recall the definition of indirect sensitivity

$$IS_Q(\mathbf{I}) = \max_{t_P \in \mathbf{I}(R_P)} S_Q(\mathbf{I}, t_P).$$

It is irrelevant to the projection operator and always increases when tuples are added. Therefore, we can use indirect sensitivity to extend the measurement $G(\cdot, \tau)$ to arbitrary SPJA queries.

For any $\tau \in \mathbb{N}$, let $\bar{F}(\mathbf{I}, \tau)$ denote the maximum size of the primary private relation instance over any $\mathbf{I}' \subseteq \mathbf{I}$ whose indirect sensitivity is bounded by τ—that is,

$$\bar{F}(\mathbf{I}, \tau) = \max_{\mathbf{I}'' \subseteq \mathbf{I}, IS_Q(\mathbf{I}'') \le \tau} |\mathbf{I}''(R_P)|,$$

and

$$\bar{G}(\mathbf{I}, \tau) = \bar{F}(\mathbf{I}, \tau) - N.$$

With a similar proof as before, we can show the following lemma.

Lemma 6.10. *For any $\mathbf{I}$ and any τ, $\bar{G}(\mathbf{I}, \tau) \le 0$ and if $\tau \ge IS_Q(\mathbf{I})$, the $\bar{G}(\mathbf{I}, \tau) = 0$.*

Lemma 6.11. *For any $\tau \in \mathbb{N}$, $\bar{G}(\cdot, \tau)$ has global sensitivity 1.*

Then, we replaces $G(\mathbf{I}, \tau)$ with $\bar{G}(\mathbf{I}, \tau)$ in ExpOPT^2. Moreover, after selecting $\tilde{\tau}$, we truncate with the LP for SPJA queries introduced in Section 5.2 instead. Due to Lemma 6.10 and 6.11, for arbitrary SPJA query, ExpOPT^2 preserves ε-DP with a similar proof as SPA queries. The utility analysis is also similar.

Theorem 6.12. *On any instance $\mathbf{I}$, ExpSPJA returns a $\tilde{Q}(\mathbf{I})$ such that with probability at least $1-\beta$,*

$$|\tilde{Q}(\mathbf{I}) - Q(\mathbf{I})| \le \frac{24 IS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2IS_Q(\mathbf{I}))}{\beta}\right).$$

For SPJA queries, ExpOPT^2 also requires an exponential running time for computing $\bar{F}(\mathbf{I}, \tau)$ while we can further propose an algorithm that can run in polynomial time by simulating $\bar{F}(\mathbf{I}, \tau)$ with LP. It is interesting to see that LP is exactly the one for $\hat{F}(\mathbf{I}, \tau)$. That is because in $\hat{F}(\mathbf{I}, \tau)$, we simulate $DS_Q(\mathbf{I}'') \le \tau$ by $S_Q(\mathbf{I}'', t_i(\mathbf{I})) \le \tau$ for all $i \in [N]$ since $DS_Q(\mathbf{I}'') = \max_{i \in [N]} S_Q(\mathbf{I}'', t_i(\mathbf{I}))$. For SPJA queries, we have $IS_Q(\mathbf{I}'') = \max_{i \in [N]} S_Q(\mathbf{I}'', t_i(\mathbf{I}))$. Therefore, we can use $S_Q(\mathbf{I}'', t_i(\mathbf{I})) \le \tau$

for all $i \in [N]$ to simulate $IS_Q(\mathbf{I}'') \leq \tau$ accordingly. We also follow $\hat{G}(\mathbf{I}, \tau) = \hat{F}(\mathbf{I}, \tau) - N$. Due to Lemma 6.6, OPT^2 still preserves ε-DP. We finally show that OPT^2 can achieve the same error bounds as ExpOPT^2 for arbitrary SPJA query. Due to Lemma 6.6, OPT^2 preserves ε-DP with a similar proof as before. We finally show that OPT^2 can achieve the same error bounds as ExpOPT^2.

THEOREM 6.13. *On any instance* $\mathbf{I}$, *OPT*^{2} *returns a* $\tilde{Q}(\mathbf{I})$ *such that with probability at least* $1 - \beta$,

$$|\tilde{Q}(\mathbf{I}) - Q(\mathbf{I})| \leq \frac{24 IS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2IS_Q(\mathbf{I}))}{\beta}\right).$$

PROOF. Let $\{y_i^*(\mathbf{I})\}_i$, $\{z_j^*(\mathbf{I})\}_j$ be the corresponding solutions of $\hat{F}(\mathbf{I}, \tilde{\tau})$. For each $j \in [M]$, let $u_j^*(\mathbf{I}) = z_j^*(\mathbf{I}) \cdot \psi(q_j(\mathbf{I}))$. Recall $L = |\pi_{\mathbf{y}} J(\mathbf{I})|$ and $p_k(\mathbf{I})$ is the k-th result in $\pi_{\mathbf{y}} J(\mathbf{I})$. For each $k \in [L]$, let $v_k^*(\mathbf{I}) = \min(\sum_{j \in E_k(\mathbf{I})} u_j^*(\mathbf{I}), \psi(p_k(\mathbf{I})))$, where $E_k(\mathbf{I}) := \{j : p_k = \pi_{\mathbf{y}} q_j(\mathbf{I})\}$. Let $V^*(\mathbf{I}) = \sum_k v_k^*(\mathbf{I})$.

Similar to the proof of Theorem 6.8, we have with probability at least $1 - \frac{\beta}{2}$,

$$\tilde{\tau} \leq 2IS_Q(\mathbf{I}), \tag{27}$$

and

$$N - \sum_i y_i^*(\mathbf{I}) \leq \frac{18}{\varepsilon} \ln\left(\frac{4(\log(2IS_Q(\mathbf{I})) + 2)}{\beta}\right). \tag{28}$$

We next show that on the one hand,

$$Q(\mathbf{I}) - V^*(\mathbf{I}) \leq \frac{18 IS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2IS_Q(\mathbf{I}))}{\beta}\right), \tag{29}$$

and on the other hand,

$$V^*(\mathbf{I}) \leq Q(\mathbf{I}, \tilde{\tau}) \leq Q(\mathbf{I}). \tag{30}$$

Combine (29) and (30), we have

$$|Q(\mathbf{I}) - Q(\mathbf{I}, \tilde{\tau})| \leq \frac{18 IS_Q(\mathbf{I})}{\varepsilon} \ln\left(\frac{4\log(2IS_Q(\mathbf{I}))}{\beta}\right).$$

Finally, we can derive the same error bound with the same proof of Theorem 6.8.

For (29), we again prove it by increasing the values iteratively. The superscript is used to show the iteration and the original ones are regarded as the values at the iteration 0. At iteration ℓ, we increase $y_i^{*(\ell-1)}(\mathbf{I})$ to 1 for $i = \ell$. We then update $z_j^{*(\ell)}(\mathbf{I}) = \sum_{i \in D_j(\mathbf{I})} y_i^{*(\ell)}(\mathbf{I}) - |D_j(\mathbf{I})| + 1$, $u_j^{*(\ell)}(\mathbf{I}) = z_j^{*(\ell)}(\mathbf{I}) \cdot \psi(q_j(\mathbf{I}))$ and $v_k^{*(\ell)}(\mathbf{I}) = \min(\sum_{j \in E_k(\mathbf{I})} u_j^{*(\ell)}(\mathbf{I}), \psi(p_k(\mathbf{I})))$. Letting $V^{*(k)}(\mathbf{I}) = \sum_k v_k^{*(\ell)}(\mathbf{I})$, we have

$$V^{*(\ell)}(\mathbf{I}) - V^{*(\ell-1)}(\mathbf{I}) \leq IS_Q(\mathbf{I}) \cdot \left(1 - y_i^{*(\ell-1)}(\mathbf{I})\right). \tag{31}$$

After all iterations, since $y_i^{*(N)}(\mathbf{I}) = 1$ for any $i \in [N]$, we have $z_j^{*(N)}(\mathbf{I}) = 1$ for any $j \in [M]$, $u_j^{*(N)}(\mathbf{I}) = \psi(q_j(\mathbf{I}))$ for any $j \in [M]$, and $v_k^{*(N)}(\mathbf{I}) = \psi(p_k(\mathbf{I}))$ for any $j \in [L]$. Thus,

$$V^{*(N)}(\mathbf{I}) = Q(\mathbf{I}). \tag{32}$$

Combining (28), (31), and (32), we can derive (29).

For (30), we show that $\{u_j^*(\mathbf{I})\}_j$, $\{v_k^*(\mathbf{I})\}_k$ is a valid solution of the LP for $Q(\mathbf{I}, \tilde{\tau})$. Since $z_j^*(\mathbf{I}) \in [0, 1]$, we can ensure $u_j^*(\mathbf{I}) \in [0, \psi(q_j(\mathbf{I}))]$ for any $j \in [M]$. For any $i \in [N]$, $\sum_{j \in C_i(\mathbf{I})}(z_j^*(\mathbf{I}) \cdot \psi(q_j(\mathbf{I}))) \leq \tilde{\tau}$ implies $\sum_{j \in C_i(\mathbf{I})} u_j^*(\mathbf{I}) \leq \tilde{\tau}$. Moreover, as we define $v_k^*(\mathbf{I}) = \min(\sum_{j \in E_k(\mathbf{I})} u_j^*(\mathbf{I}), \psi(p_k(\mathbf{I})))$, both $v_k^*(\mathbf{I}) \leq \sum_{j \in E_k(\mathbf{I})} u_j^*(\mathbf{I})$ and $0 \leq v_k^*(\mathbf{I}) \leq \psi(p_k(\mathbf{I}))$ are satisfied for any $k \in [L]$. Therefore, $\{u_j^*(\mathbf{I})\}_j$, $\{v_k^*(\mathbf{I})\}_k$ is a valid solution of the LP of $Q(\mathbf{I}, \tilde{\tau})$ and the corresponding objective value is $V^*(\mathbf{I})$. Above all, we can derive (30) since the LP maximizes the objective function. □

7 Utility Analysis of Tao et al.

In the previous sections, we show both R2T and OPT^2 achieve an error proportional to $DS_Q(\mathbf{I})$ for each instance $\mathbf{I}$. In this section, we show that the error of the algorithm in the work of Tao et al. [52], which only works for self-join-free queries, actually performs almost the same as the Laplace mechanism. More precisely, Tao et al. [52] make truncation by tuples' sensitivity, and the algorithm to find the truncation threshold is based on an upper bound on tuple sensitivities. It is denoted as ℓ in the work of Tao et al. [52], but we observe that this is just the global sensitivity. So we denote this given upper bound as GS_Q. Note that a trivial method is to set $\tau = GS_Q$, which has no bias while the error is $O\left(GS_Q \log(1/\beta)/\varepsilon\right)$ with probability $1-\beta$. The mechanism for choosing τ in the work of Tao et al. [52] is DP, but we show in the following that it is not much better than this naive choice, on *any* instance $\mathbf{I}$. More precisely, we show that its error is $\Omega\left(GS_Q/(\log(GS_Q)\varepsilon)\right)$ with at least constant probability.

Recall the algorithm first constructs a DP-version of the query result

$$\hat{Q}(\mathbf{I}) = Q(\mathbf{I}) + Lap\left(\frac{GS_Q}{\varepsilon}\right).$$

Based on this, we can see

$$\Pr[\hat{Q}(\mathbf{I}) \geq Q(\mathbf{I}) + GS_Q/\varepsilon] = \frac{1}{2e}.$$

Then, recall $Q(\mathbf{I}, \tau)$ is the query result after truncating the tuples with sensitivity larger than τ, and $Q(\mathbf{I}, \tau) \leq Q(\mathbf{I})$. Then, when $\hat{Q}(\mathbf{I}) \geq Q(\mathbf{I}) + GS_Q/\varepsilon$, for any $\tau < GS_Q/(6\ln(GS_Q/\beta))$,

$$\begin{aligned}
&\Pr[Q(\mathbf{I}, \tau) + Lap(2\tau/\varepsilon) + Lap(4\tau/\varepsilon) \geq \hat{Q}(\mathbf{I})] \\
\leq&\Pr[Q(\mathbf{I}) + Lap(2\tau/\varepsilon) + Lap(4\tau/\varepsilon) \geq \hat{Q}(\mathbf{I})] \\
\leq&\Pr[Lap(2\tau/\varepsilon) \geq GS_Q/(3\varepsilon)] + \Pr[Lap(4\tau/\varepsilon) \geq 2GS_Q/(3\varepsilon)] \\
\leq&\frac{\beta}{GS_Q}.
\end{aligned}$$

By a union bound, the SVT stops before $\tau = GS_Q/(6\ln(GS_Q/\beta))$ with probability less than β. Above all, with probability at least $\frac{1}{2e} - \beta$, the truncation threshold selected is at least $GS_Q/(6\ln(GS_Q/\beta))$.Denote E as the event $\tau \geq GS_Q/(6\ln(GS_Q/\beta))$ and $\Pr[E] \geq \frac{1}{2e} - \beta$. Then, we have

$$\begin{aligned}
&\Pr[|Q(\mathbf{I}, \tau) + Lap(\tau/\varepsilon) - Q(\mathbf{I})| \geq GS_Q/(6\varepsilon\ln(GS_Q/\beta))|E] \\
\geq&\Pr[Q(\mathbf{I}, \tau) + Lap(\tau/\varepsilon) \leq Q(\mathbf{I}) - GS_Q/(6\varepsilon\ln(GS_Q/\beta))|E] \\
\geq&\Pr[Lap(\tau/\varepsilon) \leq -GS_Q/(6\varepsilon\ln(GS_Q/\beta))|E] && (33)\\
\geq&\frac{1}{2e}. && (34)
\end{aligned}$$

As such, (33) is because, for any τ, $Q(\mathbf{I}, \tau) \leq Q(\mathbf{I})$.

Above all,

$$\begin{aligned}
&\Pr[|Q(\mathbf{I}, \tau) + Lap(\tau/\varepsilon) - Q(\mathbf{I})| \geq GS_Q/(6\varepsilon\ln(GS_Q/\beta))] \\
\geq&\Pr[|Q(\mathbf{I}, \tau) + Lap(\tau/\varepsilon) - Q(\mathbf{I})| \geq GS_Q/(6\varepsilon\ln(GS_Q/\beta))|E] \times \Pr[E] \\
\geq&\left(\frac{1}{2e} - \beta\right)\frac{1}{2e}.
\end{aligned}$$

By setting $\beta = \frac{1}{4e}$, with probability at least $\frac{1}{8e^2}$, we have

$$|M(\mathbf{I}) - Q(\mathbf{I})| \geq GS_Q/(6\varepsilon \ln(4eGS_Q)).$$

Note that this analysis holds for every instance $\mathbf{I}$, namely the mechanism in the work of Tao et al. [52] adds the same amount of noise to all instances, which equals the worst-case noise (ignoring a logarithmic factor).

8 Multiple Primary Private Relations

Now we consider the case with $k \geq 2$ primary private relations $R_P^1, \ldots, R_P^k$. In this case, two instances are considered neighbors if one can be obtained from the other by deleting a set of tuples, all of which reference the same tuple that belongs to some $R_P^i, i \in [k]$. We reduce it to the case with only one primary private relation as follows. Add a new column `ID` to every $\mathbf{I}(R_P^i), i \in [k]$, and assign unique identifiers to all tuples in these relations. Next, we construct a new relation $R_P(\mathtt{ID})$, whose physical instance $\mathbf{I}(R_P)$ consists of all these identifiers. For each R_P^i, we add an FK constraint from its `ID` column to reference the `ID` column of R_P. Note that this FK reference relationship is actually a bijection between the `ID` column in R_P and all identifiers in the primary private relations. Now, we designate R_P as the only primary private relation, whereas $R_P^i, i \in [k]$ all become secondary private relations. The original secondary private relations (i.e., those having FK references to the R_P^i's directly or indirectly) are still secondary private relations.

It is not hard to see that (1) the query answer is not affected by this schema change; (2) two instances in the original schema are neighbors if and only if they are neighbors in the new schema; and (3) the join results that reference any tuple $t \in \mathbf{I}(R_P^i), i \in [k]$ are the same as those that reference $t_P \in \mathbf{I}(R_P)$, where t_P and t have the same identifier. Thus, both the privacy and utility guarantees of our algorithm continue to hold.

Finally, it is worth pointing out that the preceding reduction is conceptual; in the actual implementation, there is no need to construct the new primary private relation and the additional `ID` columns, as illustrated in Example 9.1 of the next section.

9 System Implementation

Based on R2T and OPT^2, we have implemented two systems on top of PostgreSQL and CPLEX. The system structures respectively are shown in Figures 3 and 4. The input to our system is any SPJA query written in SQL, together with a designated primary private relation R_P (interestingly, while the mechanisms satisfy the DP policy with FK constraints, the algorithm itself does not need to know the PK-FK constraints).

Both two systems support `SUM` and `COUNT` aggregation, and they share a similar process. For both of them, our SQL parser first unpacks the aggregation into a reporting query so as to find $\psi(q_j(\mathbf{I}))$ for each join result, as well as $C_i(\mathbf{I})$ and $D_j(\mathbf{I})$, which store the referencing relationships between tuples in $\mathbf{I}(R_P)$ and $J(\mathbf{I})$.

Example 9.1. Suppose we use the TPC-H schema (shown in Figure 5), where we designate `Supplier` and `Customer` as primary private relations. Consider the following query.

```
SELECT SUM(price * (1 − discount))
FROM Supplier, Lineitem, Orders, Customer
WHERE Supplier.SK = Lineitem.SK AND Lineitem.OK = Orders.OK
   AND Orders.CK = Customer.CK AND Orders.orderdate >=′ 2020 − 08 − 01′
```

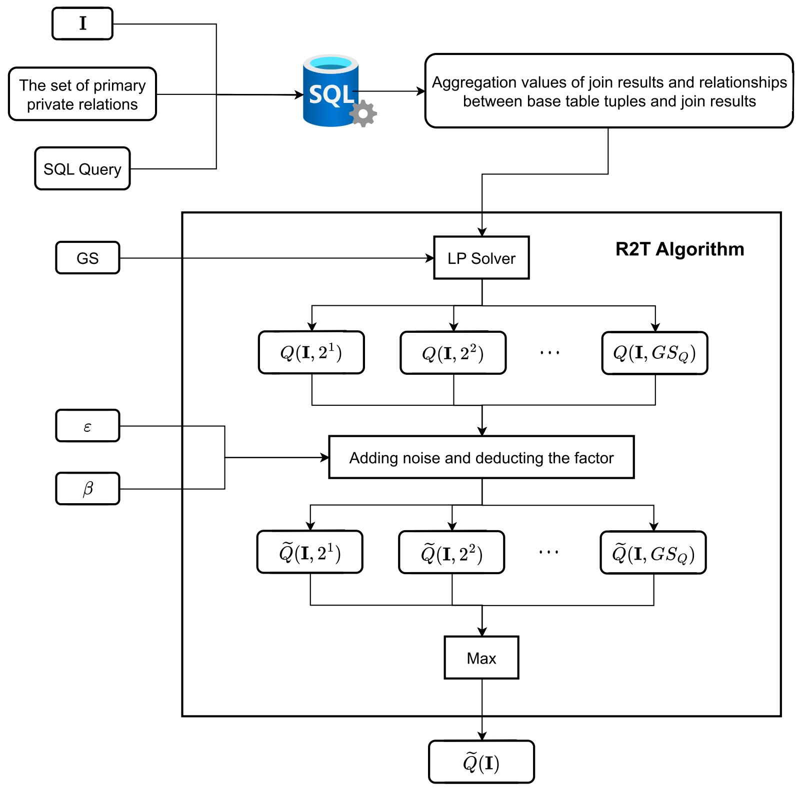

Fig. 3. System structure for R2T.

We rewrite it as follows.

```
SELECT Supplier.SK, Customer.CK, price * (1 − discount)
FROM Supplier, Lineitem, Orders, Customer
WHERE Supplier.SK = Lineitem.SK AND Lineitem.OK = Orders.OK
   AND Orders.CK = Customer.CK AND Orders.orderdate >=' 2020 − 08 − 01'
```

The `price * (1 − discount)` column in the query results gives all the $\psi(q_j(\mathbf{I}))$ values, whereas `Supplier.SK` and `Customer.CK` yield the referencing relationships from each supplier and customer to all the join results to which they contribute. □

We execute the rewritten query in PostgreSQL and export the query results to a file. Next, we feed the results to CPLEX and then combine the LP results as indicated by our mechanisms to obtain the privatized output. More precisely, R2T and OPT^2 compute $Q(\mathbf{I}, \tau)$ and $\hat{F}(\mathbf{I}, \tau)$ across values of $\tau = 2, 4, 8, \ldots, GS_Q$ respectively, each of which is derived from solving an LP. Notice that LP formulation of $\hat{F}(\mathbf{I}, \tau)$ is more complex than $Q(\mathbf{I}, \tau)$. To compute the final output, R2T adopts a straightforward approach, applying noise directly to each $Q(\mathbf{I}, \tau)$ and selecting the maximum for the output. In contrast, OPT^2 incorporates a series of more elaborate steps. It first inputs $\hat{F}(\mathbf{I}, \tau)$'s into an SVT to determine an appropriate truncation threshold $\tilde{\tau}$. Then, OPT^2 executes an LP to compute the truncated result, which is finally outputted after adding a noise proportional to $\tilde{\tau}$. Overall, OPT^2 involves more complex computational steps compared to R2T, but as mentioned, OPT^2 yields a better utility in theory.

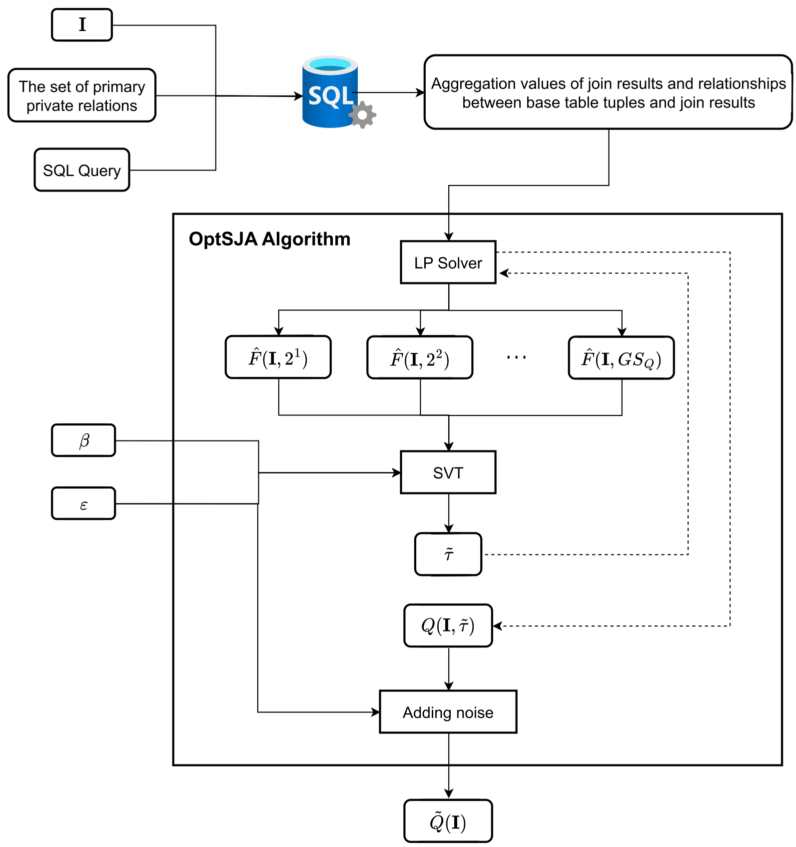

Fig. 4. System structure for OPT^2.

Besides, for both mechanisms, the computation bottleneck is the LPs. This takes polynomial time but can still be very expensive in practice. One straightforward optimization is to solve them in parallel. In the following, we further present an effective technique, which can be used to speed up this process for both R2T and OPT^2.

9.1 Early Stop

Early Stop for R2T. For R2T, the key observation is that it returns the maximum of $O\left(\log(GS_Q)\right)$ maximization LPs (masked by some noise and reduced by a factor), and most LP solvers (e.g., CPLEX) for maximization problems use some iterative search technique to gradually approach the optimum from below—namely, these $O\left(\log(GS_Q)\right)$ LP solvers all "race to the top." Thus, we will not know the winner until they all stop.

To cut down the unnecessary search, the idea is to flip the problem around. Instead of solving the primal LPs, we solve their duals. By LP duality, the dual LP has the same optimal solution as the primal, but importantly, the LP solver will approach the optimal solution from above—namely, we have a gradually decreasing upper bound for the optimal solution of each LP. This allows us to terminate those LPs that have no hope to be the winner. The optimized R2T algorithm, shown in Algorithm 3, also uses the trick that the noises are generated before we start running the LP solvers so that we know when to terminate.

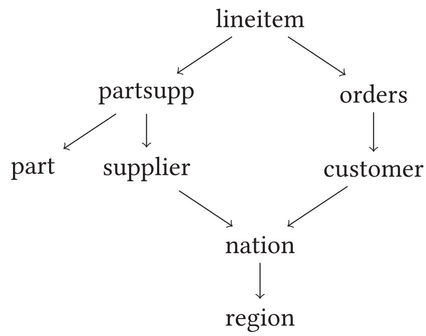

Fig. 5. The FK graph of the TPC-H schema.

ALGORITHM 3: R2T with early stop

Input: $\mathbf{I}$, ε, β, Q,R_P, GS_Q

1 $\tilde{Q}(\mathbf{I}) \leftarrow 0$;
2 **for** $\tau^{(\ell)} \leftarrow GS_Q, GS_Q/2, \dots, 2$ **do in parallel**
3 $\quad v^{(\ell)} \leftarrow Lap\left(\log(GS_Q)\frac{\tau^{(\ell)}}{\varepsilon}\right) - \log(GS_Q)\ln\left(\frac{\log(GS_Q)}{\beta}\right)\cdot\frac{\tau^{(\ell)}}{\varepsilon}$;
4 $\quad$ **for** $k \leftarrow 1, 2, \dots$ **do**
5 $\quad\quad$ **if** $\hat{Q}^{(k)}(\mathbf{I}, \tau^{(\ell)})$ *achieves the optimal* **then**
6 $\quad\quad\quad \tilde{Q}(\mathbf{I}) \leftarrow \max(\tilde{Q}(\mathbf{I}), \hat{Q}^{(k)}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)})$;
7 $\quad\quad\quad$ Break;
8 $\quad\quad$ **else if** $\hat{Q}^{(k)}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)} \leq \tilde{Q}(\mathbf{I})$ **then**
9 $\quad\quad\quad$ Break;
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 **end**
13 **return** $\tilde{Q}(\mathbf{I})$;

In Algorithm 3, we use k to denote the iteration of the LP solver and use $\hat{Q}^{(k)}(\mathbf{I}, \tau)$ to denote the solution to the dual LP at the k-th iteration. A technicality is that in line 1, we should initialize $\tilde{Q}(\mathbf{I})$ to $Q(\mathbf{I}, 0)$ to be consistent with the R2T algorithm, but $Q(\mathbf{I}, 0) = 0$ for all truncation methods described in this article.

When there are not enough CPU cores to solve all LPs in parallel, we choose to start with those with a larger τ in line 3 of Algorithm 3. This is based on our observation that those LPs tend to terminate faster. This is quite intuitive: when τ is larger, the optimal solution is also higher, thus the LP solver for the dual can terminate earlier.

Early Stop for OPT^2. For OPT^2, we need to solve $O\left(\log(DS_Q(\mathbf{I}))\right)$ number of LPs for $F(\mathbf{I}, \tau)$'s and one LP for $Q(\mathbf{I}, \tilde{\tau})$. While we need the exact value for the last LP for $Q(\mathbf{I}, \tilde{\tau})$, we can apply the similar "early stop" idea to the LPs for $F(\mathbf{I}, \tau)$'s.

The algorithm is shown in Algorithm 4, where k is the iteration of the LP solver, and $\hat{F}^{(k)}(\mathbf{I}, \tau)$ denotes the solution to the dual LP at the k-th iteration. Note that as we define $\hat{G}(\mathbf{I}, \tau) = \hat{F}(\mathbf{I}, \tau) - N$, the algorithm SVT$(-9\ln(4/\beta)/\varepsilon, 2\varepsilon/3, \hat{G}(\mathbf{I}, 2), \hat{G}(\mathbf{I}, 4), \hat{G}(\mathbf{I}, 8), \dots)$ is exactly the same as the algorithm SVT$(N - 9\ln(4/\beta)/\varepsilon, 2\varepsilon/3, \hat{F}(\mathbf{I}, 2), \hat{F}(\mathbf{I}, 4), \hat{F}(\mathbf{I}, 8), \dots)$.

In Algorithm 4, we use $\tilde{\ell}$ to store the candidate output of the SVT and initialize it to $+\infty$. We first compute $\tilde{T}$ as in SVT and generate all noises before solving the LPs. Then, at each iteration

ALGORITHM 4: OPT^2 with early stop

Input: $\mathbf{I}, \varepsilon, \beta, Q, R_P$

$\tilde{\ell} \leftarrow +\infty$;
$T \leftarrow N - 9\ln(4/\beta)/\varepsilon$;
$\tilde{T} \leftarrow T + Lap(3/\varepsilon)$;
for $\ell \leftarrow 1, 2, \ldots$ **do in parallel**
  $\tau^{(\ell)} \leftarrow 2^{\ell}$;
  $v^{(\ell)} \leftarrow Lap(6/\varepsilon)$;
  for $k \leftarrow 1, 2, \ldots$ **do**
    if $\ell \geq \tilde{\ell}$ **then**
      Break;
    else if $\hat{F}^{(k)}(\mathbf{I}, \tau^{(\ell)})$ *achieves the optimal* **then**
      if $\hat{F}^{(k)}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)} > \tilde{T}$ **then**
        if $\ell \leq \tilde{\ell}$ **then**
          $\tilde{\ell} \leftarrow \ell$;
        Break;
      end
    end
    else if $\hat{F}^{(k)}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)} \leq \tilde{T}$ **then**
      Break;
    end
  end
end
$\tilde{\tau} \leftarrow 2^{\tilde{\ell}}$;
$\tilde{Q}(\mathbf{I}) \leftarrow Q(\mathbf{I}, \tilde{\tau}) + Lap\left(\frac{3\tilde{\tau}}{\varepsilon}\right)$;
return $\tilde{Q}(\mathbf{I})$;

k, if the LP achieves the optimal, we check whether the SVT would stop, and if so, we store the corresponding ℓ as the candidate output. Otherwise, if $\hat{F}^{(k)}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)} \leq \tilde{T}$, since the LP solver approaches the optimal solution from above, we know that $\hat{F}(\mathbf{I}, \tau^{(\ell)}) + v^{(\ell)} < \tilde{T}$, thus we can terminate this LP. Moreover, when we get some candidate output $\tilde{\ell}$, we directly terminate all LPs $\hat{F}(\mathbf{I}, \tau^{(\ell)})$'s with $\ell \geq \tilde{\ell}$ since the SVT must stop at or before $\tilde{\ell}$.

The improvement brought by the early stop technique is that we reduce the computation of $\hat{F}(\mathbf{I}, \tau)$ for some small τ's. It seems this step can only reduce the computation by a constant factor; however, the improvement is large in practice: the computational cost of $\hat{F}(\mathbf{I}, \tau)$ increases largely as τ decreases. This is intuitive. Recall τ is the constraint for the contribution of each user and $\hat{F}(\mathbf{I}, \tau)$ maximizes the number of users. When τ is large, most constraints are satisfied automatically, and thus we can search for the optimal solution fast. This also matches the case of R2T.

10 Experiments

We conducted experiments on two types of queries: graph pattern counting queries under node-DP and general SPJA queries with FK constraints, with the former being an important special case of the latter. For graph pattern counting queries, we compare R2T and OPT^2 with naive truncation with smooth sensitivity (NT) [33], the smooth distance estimator (SDE) [9], the recursive mechanism (RM) [11], and the LP-based mechanism (LP) [33]. For general SPJA queries, we compare with the local sensitivity-based mechanism (LS) [52].

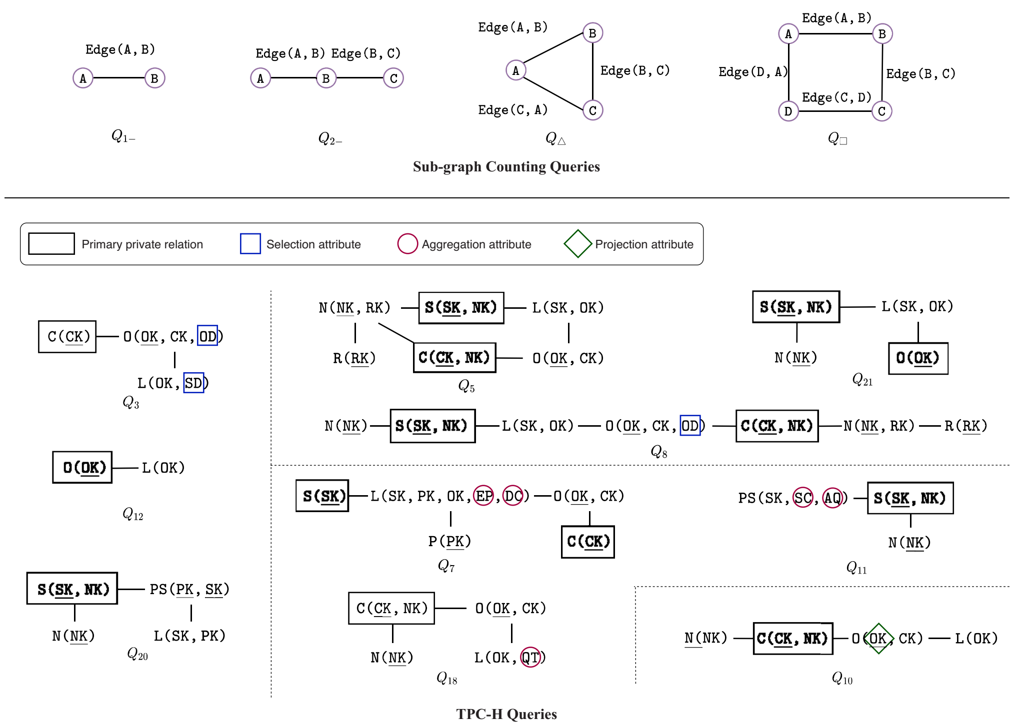

Fig. 6. The structure of queries.

Table 1. Graph Datasets Used in the Experiments

Dataset	**Deezer**	**Amazon1**	**Amazon2**	**RoadnetPA**	**RoadnetCA**
Nodes	144,000	262,000	335,000	1,090,000	1,970,000
Edges	847,000	900,000	926,000	1,540,000	2,770,000
Maximum degree	420	420	549	9	12
Degree upper bound D	1,024	1,024	1,024	16	16

10.1 Setup

Queries. For graph pattern counting queries, we used four queries: edge counting Q_{1-}, length-2 path counting Q_{2-}, triangle counting $Q_{\triangle}$, and rectangle counting $Q_{\square}$. For SPJA queries, we used 10 queries from the TPC-H benchmark, whose structures are shown in Figure 6. These queries involve a good mix of selection, projection, join, and aggregation. We removed all the group-by clauses from the queries—a brief discussion on this is provided at the end of the article.

Datasets. For graph pattern counting queries, we used 5 real world networks datasets: **Deezer**, **Amazon1**, **Amazon2**, **RoadnetPA**, and **RoadnetCA**. **Deezer** collects the friendships of users from the music streaming service Deezer. **Amazon1** and **Amazon2** are two Amazon co-purchasing networks. **RoadnetPA** and **RoadnetCA** are road networks of Pennsylvania and California, respectively. All these datasets are obtained from SNAP [35]. Table 1 shows the basic statistics of these datasets.

Most algorithms need to assume a GS_Q in advance. Note that the value of GS_Q should not depend on the instance but may use some background knowledge for a particular class of instances. Thus, for the three social networks, we set a degree upper bound of $D = 1024$, whereas for the two road networks, we set $D = 16$. Then we set GS_Q as the maximum number of graph patterns containing any node. This means $GS_{Q_{1-}} = D$, $GS_{Q_{2-}} = GS_{Q_{\triangle}} = D^2$, and $GS_{Q_{\square}} = D^3$. For TPC-H queries, we used datasets of scale $2^{-3}, 2^{-2}, \ldots, 2^3$. The one with scale 1 (default scale) has about 7.5 million tuples, and we set $GS_Q = 10^6$.

The LP mechanism requires a truncation threshold τ, but Kasiviswanathan et al. [33] do not discuss how this should be set. Initially, we used a random threshold uniformly chosen from $[1, GS_Q]$. This turned out to be very bad, as with constant probability, the picked threshold is $\Omega(GS_Q)$, which makes these mechanisms as bad as the naive mechanism that adds GS_Q noise. To achieve better results, as in R2T, we consider $\{2, 4, 8, \ldots, GS_Q\}$ as the possible choices. Similarly, NT and SDE need a truncation threshold θ on the degree, and we choose one from $\{2, 4, 8, \ldots, D\}$ randomly.

Experimental Environment. All experiments were conducted on a Linux server with a 24-core 2.2GHz Intel Xeon CPU and 256GB of memory. Each program was allowed to use at most 10 threads, and we set a time limit of 6 hours for each run. Each experiment was repeated 20 times, and we report the average running time. The errors are less stable due to the random noise, so we remove the best 4 and worst 4 runs, and report the average error of the remaining 12 runs. The failure probability β in R2T is set to 0.1. The default DP parameter is $\varepsilon = 0.8$.

10.2 Graph Pattern Counting Queries

Utility and Efficiency. The errors and running times of all mechanisms over the graph pattern counting queries are shown in Table 2. These results indicate a clear superiority of R2T and OPT^2 in terms of utility, offering order-of-magnitude improvements over other methods in many cases. What is more desirable is its robustness: in all 20 query-dataset combinations, R2T consistently achieves an error below 20%, whereas the error is below 10% in all but 3 cases. Meanwhile, OPT^2 can be computed within a 6-hour time limit in 17 cases, where the error is always below 10% and is below 1% in all but 4 cases. We also notice that, given a query, R2T and OPT^2 perform better in road networks than social networks. This is because their errors are proportional to $DS_Q(\mathbf{I})$ by our theoretical analysis. Thus, the relative error is proportional to $DS_Q(\mathbf{I})/|Q(\mathbf{I})|$. Therefore, larger and sparser graphs, such as road networks, lead to smaller relative errors. Besides, OPT^2 always has a smaller error than R2T and the gap can be as large as 20x, which confirms our theoretical analysis that OPT^2 has a smaller optimality ratio than R2T.

In terms of running time, all mechanisms are reasonable except RM. RM can only complete within the 6-hour time limit on three cases, although it achieves small errors on these three cases. In addition, SDE is faster than RM but runs a bit slower than others in most cases. OPT^2 has good efficiency in all query-dataset combinations except five cases where the running time is over 1 hour. Furthermore, compared with R2T, OPT^2 always has much more running time. That is because, even though OPT^2 and R2T solve $O(\log DS_Q(\mathbf{I}))$ and $O(\log GS_Q)$ LPs, respectively, the LPs for OPT^2 are a bit more complex than R2T. Overall, OPT^2 and R2T do not dominate each other. OPT^2 has higher utility while R2T has higher efficiency. Besides, another interesting observation is R2T sometimes even runs faster than LP, despite the fact that R2T needs to solve $O(\log GS_Q)$ LPs. This is due to the early stop optimization: the running time of R2T is determined by the LP that corresponds to the near-optimal τ, which often happens to be one of the LPs that can be solved fastest.

Privacy Parameter ε. Next, we conducted experiments to see how the privacy parameter ε affects various mechanisms. We tested different queries on **RoadnetPA** where we vary ε from 0.1 to 12.8. We plot the results in Figure 7, where we also plot the query result to help see the utilities of the mechanisms. The first message from the plot is the same as before, that all R2T, OPT^2, and RM achieve high utility (but RM spends 200x more time). NT and SDE lose utility (i.e., error larger

Table 2. Comparison among R2T, OPT^2, Naive Truncation with Smooth Sensitivity (NT), the Smooth Distance Estimator (SDE), the LP-Based Mechanism (LP), and the Recursive Mechanism (RM) on Graph Pattern Counting Queries

Dataset			**Deezer**	**Amazon1**	**Amazon2**	**RoadnetPA**	**RoadnetCA**
q_{1-}	Query result		847,000	900,000	926,000	1,540,000	2,770,000
	Query Running Time(s)		1.28	1.52	1.62	1.51	2.64
	R2T	Relative error(%)	0.535	0.557	0.432	0.0114	0.00635
		Time(s)	12.3	15.6	16.2	26.8	48.7
	OPT^2	Relative error(%)	0.327	0.329	0.218	0.00127	0.000932
		Time(s)	165	194	209	223	411
	NT	Relative error(%)	59.1	101	125	1,370	1,410
		Time(s)	18.1	29.3	40.4	21.9	39.7
	SDE	Relative error(%)	548	363	286	55.2	81.8
		Time(s)	9,870	4,570	1,130	105	292
	LP	Relative error(%)	14.3	5.72	6.75	3.6	3.02
		Time(s)	16.9	14.7	14.4	28.3	54
q_{2-}	Query result		21,800,000	9,120,000	9,750,000	3,390,000	6,000,000
	Query Running Time(s)		13.8	11.8	13.8	6.39	6.06
	R2T	Relative error(%)	6.64	12.2	9.06	0.0539	0.0352
		Time(s)	356	170	196	80.2	145
	OPT^2	Relative error(%)	Over time limit		4.66	0.0108	0.00166
		Time(s)			20,900	830	1,800
	NT	Relative error(%)	116	398	390	6,160	6,530
		Time(s)	21.0	28.4	41.0	23.2	44.2
	SDE	Relative error(%)	8,900	5,110	1,930	211	228
		Time(s)	9,870	4,570	1,130	104	296
	LP	Relative error(%)	35.9	23.2	27.8	11.1	13.3
		Time(s)	8,820	3,600	461	148	404
$q_{\triangle}$	Query result		794,000	718,000	667,000	67,200	121,000
	Query Running Time(s)		4.53	5.03	4.20	2.96	5.17
	R2T	Relative error(%)	5.58	1.27	2.03	0.102	0.061
		Time(s)	17.3	18.8	19.9	4.21	7.5
	OPT^2	Relative error(%)	1.05	0.542	0.359	0.0245	0.0258
		Time(s)	334	225	225	5.74	11.3
	NT	Relative error(%)	782	1,660	1,920	110,000	105,000
		Time(s)	23.0	31.7	41.0	23.3	45.0
	SDE	Relative error(%)	67,300	26,000	9,600	4,150	3,830
		Time(s)	9,880	4,570	1,130	106	297
	LP	Relative error(%)	24.6	12.8	14.2	0.104	0.0625
		Time(s)	131	18.2	18.3	3.95	7.06
	RM	Relative error(%)	Over time limit			0.0388	0.0193
		Time(s)				1,280	2,550
$q_{\square}$	Query result		11,900,000	2,480,000	3,130,000	158,000	262,000
	Query Running Time(s)		74.3	21.6	15.6	4.50	10.1
	R2T	Relative error(%)	16.9	6.29	10.5	0.0729	0.0638
		Time(s)	289	70.5	86.8	8.18	16.2
	OPT^2	Relative error(%)	Over time limit	2.36	2.48	0.0166	0.00599
		Time(s)		1,920	4,510	28.8	46.1
	NT	Relative error(%)	3,750	30,700	26,100	319,000	368,000
		Time(s)	57.6	35.8	50.6	24.8	45.0
	SDE	Relative error(%)	6,970,000	11,400,000	202,000	10,300	9,130
		Time(s)	9,930	4,580	1,140	108	300
	LP	Relative error(%)	92.6	94.8	77.8	0.223	0.165
		Time(s)	2,530	70.4	81.2	7.83	14.2
	RM	Relative error(%)	Over time limit			0.0217	Over time limit
		Time(s)				10,500	

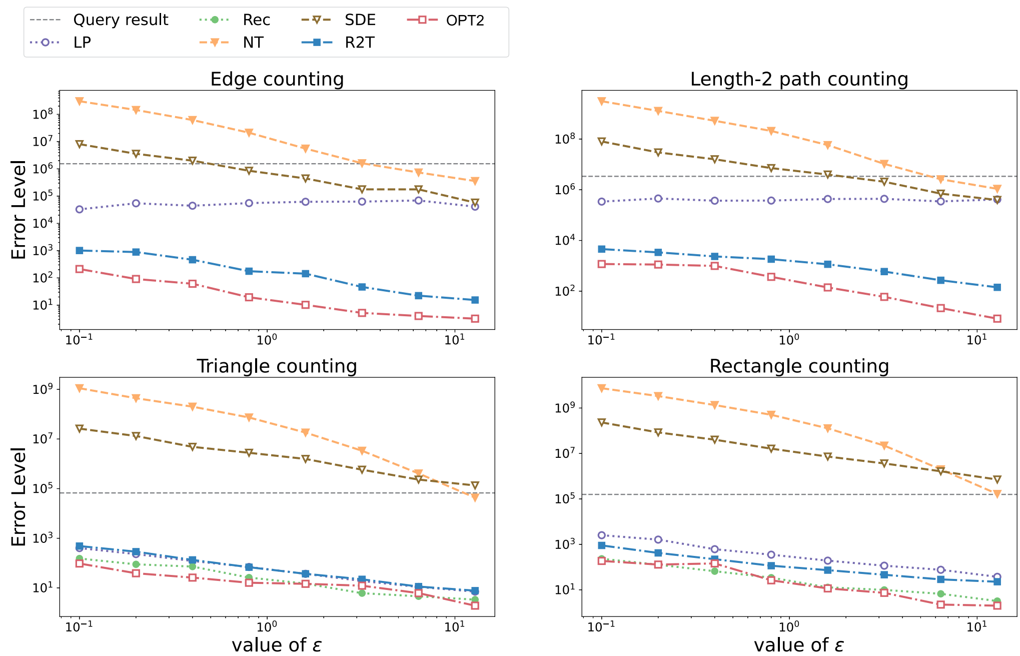

Fig. 7. Error levels of various mechanisms on graph pattern counting queries over **RoadnetPA** with various values of ε.

Table 3. Error Levels of R2T, OPT^2, and the LP-Based Mechanism (LP) with Different τ for Queries on **Amazon2**

Query		Q_{1-}	Q_{2-}	$Q_{\triangle}$	$Q_{\square}$
Query result		926,000	9,750,000	667,000	3,130,000
R2T		4,000	883,000	13,500	328,000
OPT^2		2020	455,000	2,400	77,400
LP	$\tau = GS_Q$	1,440	1,580,000	1,290,000	1,370,000,000
	$\tau = GS_Q/8$	2,100	181,000	157,000	140,000,000
	$\tau = GS_Q/64$	110,000	259,000	15,100	25,800,000
	$\tau = GS_Q/512$	645,000	1,260,000	2,790	2,630,000
	$\tau = GS_Q/4096$	810,000	3,950,000	2,090	274,000
	$\tau = GS_Q/32768$	911,000	7,580,000	92,300	48,700
	$\tau = GS_Q/262144$	924,000	9,340,000	459,000	76,400
	Average error	62,500	2,710,000	94,900	2,430,000

than query result) except for very large ε. LP achieves similar utility as R2T and OPT^2 on $Q_{\triangle}$ and $Q_{\square}$, but it is much worse on Q_{1-} and Q_{2-}. In particular, a higher ε does not help LP on these two queries, because the bias (further controlled by a randomly selected τ) dominates the error for these two queries.

Selection of τ. In the next set of experiments, we dive deeper and see how sensitive the utility is with respect to the truncation threshold τ. We tested the queries on **Amazon2** and measured the error of the LP-based mechanism [33] with different τ. For each query, we tried various τ from 2 to GS_Q and compared their errors with R2T. The results are shown in Table 3, where the optimal error

Table 4. Running Times of R2T and OPT^2 for $Q_\square$ with and without Early Stop

Dataset			**Deezer**	**Amazon1**	**Amazon2**	**RoadnetPA**	**RoadnetCA**
Time(s)	R2T	With early stop	289	70.5	86.8	8.18	16.2
		Without early stop	28,700	537	422	12.8	16.4
		Speedup	99.3×	7.62 ×	4.86×	1.56×	1.01×
	OPT^2	With early stop	Over time limit	1,920	4,510	28.8	46.1
		Without early stop		$\geq 384,000$	$\geq 451,000$	49	101
		Speedup		$\geq$ 200×	$\geq$ 100×	1.7×	2.19×

Table 5. Comparison between R2T and the Local-Sensitivity-Based Mechanism (LS) on SQL Queries

Query			Query result	R2T	OPT^2	LS
Single primary private relation	Q_3	Value/Relative error(%)	2,890,000	0.254	0.0108	38.8
		Time(s)	1.6	18.9	303	19.2
	Q_{12}	Value/Relative error(%)	6,000,000	0.0229	0.000345	16.3
		Time(s)	1.24	28.2	194	25.8
	Q_{20}	Value/Relative error(%)	6,000,000	0.579	0.0454	15.4
		Time(s)	1.25	24.5	648	24.4
Multiple primary private relation	Q_5	Value/Relative error(%)	240,000	1.626	0.089	Not supported
		Time(s)	2.51	8.42	36.2	
	Q_8	Value/Relative error(%)	1,830,000	1.92	0.0336	
		Time(s)	1.41	39.6	497	
	Q_{21}	Value/Relative error(%)	6,000,000	0.654	0.0397	
		Time(s)	2.32	124	2950	
Sum aggregation	Q_7	Value/Relative error(%)	218,000,000	0.607	0.0555	
		Time(s)	3.22	140	3580	
	Q_{11}	Value/Relative error(%)	2,000,000	1.82	0.0253	
		Time(s)	0.29	4.41	82.9	
	Q_{18}	Value/Relative error(%)	153,000,000	0.132	0.00826	
		Time(s)	2.21	42.7	1220	
Projection	Q_{10}	Value/Relative error(%)	1,500,000	0.174	0.0341	
		Time(s)	0.32	8.77	684	

is marked in gray. The results indicate that the error is highly sensitive to τ, and more importantly, the optimal choice of τ closely depends on the query, and there is no fixed τ that works for all cases. However, the errors of R2T and OPT^2 are within a small constant factor (around 6 for R2T and 2 for OPT^2) to the optimal choice of τ, which is exactly the value of instance optimality.

Early Stop Optimization. We also did some experiments to compare the running time of R2T with and without the early stop optimization. Here, we ran $Q_\square$ over different datasets and the results are shown in Table 4. From this table, we can see, for both R2T and OPT^2, the early stop is particularly useful in cases with long running times. In these cases, one or two LPs, which do not correspond to the optimal choice of τ, take a long time to run, and early stop is able to terminate these LPs as soon as possible.

10.3 SPJA Queries

Utility and Efficiency. We tested 10 queries from the TPC-H benchmark comparing R2T, OPT^2, and LS, and the results are shown in Table 5. We see that both R2T and OPT^2 achieve

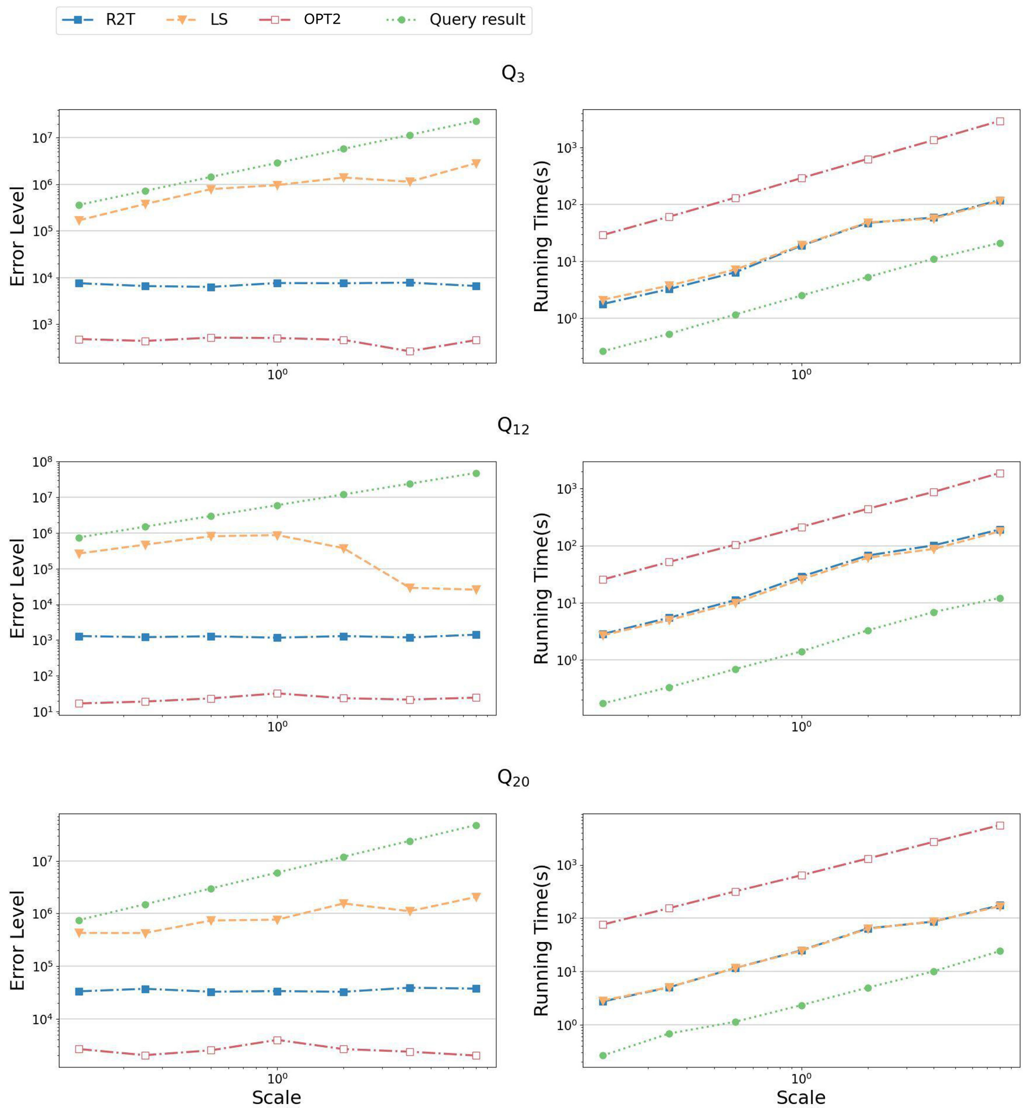

Fig. 8. Running times and error levels of R2T and the local-sensitivity based mechanism (LS) for different data scales.

order-of-magnitude improvements over LS in terms of utility. More importantly, R2T and OPT^2 support a variety of SPJA queries that are not supported by LS, with robust performance across the board. Besides, compared with graph pattern counting queries, the gap of error between R2T and OPT^2 is a bit larger, which can be as large as 100x. That is because we use larger GS_Q in TPC-H queries and OPT^2 mainly reduces a $\log(GS_Q)$ factor in error compared with R2T.

Scalability. To examine the effects as the data scale changes, we used TPC-H datasets with scale factors ranging from 2^{-3} to 2^3 with Q_3, Q_{12}, and Q_{20}. We compare both the errors and running times of R2T, OPT^2, and LS. The results are shown in Figure 8. From the results, we see that the errors of R2T and OPT^2 barely increase with the data size. The reason is that our errors only depend on

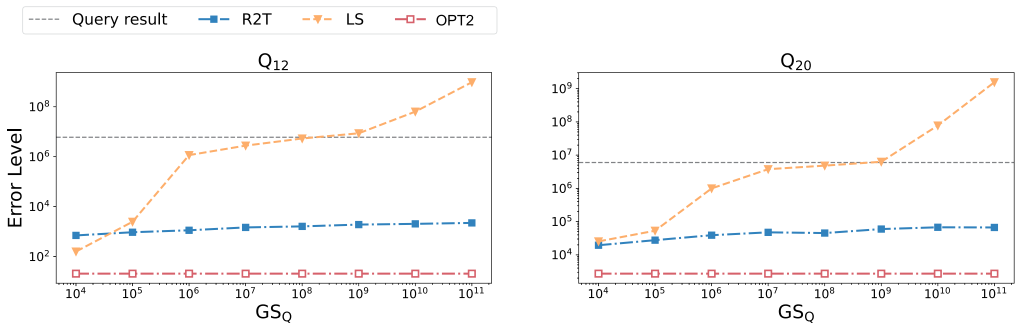

Fig. 9. Error levels of R2T and local-sensitivity based mechanism (LS) with different GS_Q.

$DS_Q(\mathbf{I})$, which does not change much by the scale of TPC-H data. However, the behavior of LS is more complicated. For Q_3 and Q_{20}, its error increases with the data size; for Q_{12}, its error increases first but then decreases later. This is because LS runs an SVT on the sensitivities of tuples to choose τ, which is closely related to the distribution of tuples' sensitivities. This is another indication that selecting a near-optimal τ is not an easy task. In terms of running time, both mechanisms have the running time linearly increase with the data size, which is expected.

Dependency on GS_Q. Our last set of experiments examines the effect GS_Q brings to the utilities of R2T, OPT^2, and LS. We conducted experiments using Q_{12} and Q_{20} with different values of GS_Q. The results are shown in Figure 9. First, OPT^2 is independent on GS_Q and always achieves the smallest error. For R2T and LS, when GS_Q is small, the errors of these two mechanisms are very close. When GS_Q increases, the error of LS increases rapidly and loses the utility (error larger than query result) very soon. Meanwhile, the error of R2T increases very slowly with GS_Q. This confirms our analysis that the error of LS grows near linearly as GS_Q, whereas that of R2T grows logarithmically. The important consequence is that, with R2T, one can be very conservative in setting the value of GS_Q. This gives the DBA a much easier job, in case she/he has little idea on what datasets the database is expected to receive.

11 Additional Discussion

Following this work, there have been many efforts put into query evaluation in relational databases under DP. For instance, Dong and Yi [23] improve the logarithmic factor in the error for self-join-free queries, whereas Fang et al. [28] explore answering SPJA queries with `Max` aggregation. In addition, Cai et al. [10] and Dong et al. [18] focus on answering multiple queries, and Dong et al. [15, 17] investigate SPJA queries over dynamic databases. For more details, please refer to the recent survey of Dong and Yi [21]. Moreover, by integrating this work with other works [18, 28], we have developed a DP SQL system [56] capable of answering a broad class of queries that include selection, projection, aggregation, join, and group by operations.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. 2019. Bounding user contributions: A bias-variance trade-off in differential privacy. In *Proceedings of the International Conference on Machine Learning*. 263–271.

[3] Galen Andrew, Om Thakkar, H. Brendan McMahan, and Swaroop Ramaswamy. 2019. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871* (2019).

[4] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. 2016. Sensitivity of counting queries. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP'16)*.
[5] Hilal Asi and John C. Duchi. 2020. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in Neural Information Processing Systems* 33 (2020), 1–12.
[6] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 273–282.
[7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 464–473.
[8] Jaroslaw Błasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. 2019. Towards instance-optimal private query release. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*. 2480–2497.
[9] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. 87–96.
[10] Kuntai Cai, Xiaokui Xiao, and Graham Cormode. 2023. PrivLava: Synthesizing relational data with foreign keys under differential privacy. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.
[11] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.
[12] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. 123–138.
[13] Apple Differential Privacy Team. n.d. Learning with Privacy at Scale. Retrieved September 27, 2024 from https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf
[14] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 3574–3583.
[15] Wei Dong, Zijun Chen, Qiyao Luo, Elaine Shi, and Ke Yi. 2024. Continual observation of joins under differential privacy. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[16] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proceedings of the 2022 International Conference on Management of Data*. 759–772.
[17] Wei Dong, Qiyao Luo, and Ke Yi. 2023. Continual observation under user-level differential privacy. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP'23)*. IEEE, 2190–2207.
[18] Wei Dong, Dajun Sun, and Ke Yi. 2023. Better than composition: How to answer multiple relational queries under differential privacy. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–26.
[19] Wei Dong and Ke Yi. 2021. Residual sensitivity for differentially private multi-way joins. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
[20] Wei Dong and Ke Yi. 2022. A nearly instance-optimal differentially private mechanism for conjunctive queries. In *Proceedings of the ACM Symposium on Principles of Database Systems*.
[21] Wei Dong and Ke Yi. 2023. Query evaluation under differential privacy. *ACM SIGMOD Record* 52, 3 (2023), 6–17.
[22] Wei Dong and Ke Yi. 2023. Universal private estimators. In *Proceedings of the ACM Symposium on Principles of Database Systems*.
[23] Wei Dong and Ke Yi. 2023. Universal private estimators. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 195–206.
[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference*. 265–284.
[25] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. 2009. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09)*. ACM, New York, NY, USA, 381–390. https://doi.org/10.1145/1536414.1536467
[26] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3-4 (2014), 211–407.
[27] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1054–1067.
[28] Juanru Fang, Wei Dong, and Ke Yi. 2022. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1009–1022.

[29] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. *Advances in Neural Information Processing Systems* 25 (2012), 2339–2347.

[30] Ziyue Huang, Yuting Liang, and Ke Yi. 2021. Instance-optimal mean estimation under differential privacy. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS'21)*. 1–12.

[31] Noah Johnson, Joseph P. Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

[32] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[33] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Proceedings of the Theory of Cryptography Conference*. 457–476.

[34] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: A differentially private SQL query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.

[35] Jure Leskovec and Andrej Krevl. 2016. SNAP Datasets: Stanford Large Network Dataset Collection (2014). Retrieved September 27, 2024 from https://snap.stanford.edu/data

[36] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: Optimizing linear counting queries under differential privacy. *VLDB Journal* 24, 6 (2015), 757–781.

[37] C. Lund and M. Yannakakis. 1993. The approximation of maximum subgraph problems. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*.

[38] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets practice on the map. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. IEEE, 277–286.

[39] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).

[40] Frank D. McSherry. 2009. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. 19–30.

[41] Arjun Narayan and Andreas Haeberlen. 2012. DJoin: Differentially private join queries over distributed databases. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*. 149–162.

[42] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. 2013. The geometry of differential privacy: The sparse and approximate cases. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. 351–360.

[43] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. 75–84.

[44] Catuscia Palamidessi and Marco Stronati. 2012. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In *Proceedings of the 10th Workshop on Quantitative Aspects of Programming Languages (QAPL'12)*. 92–105.

[45] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. 2019. AdaCliP: Adaptive clipping for private SGD. *arXiv preprint arXiv:1908.07643* (2019).

[46] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating data to sensitivity in private data analysis. *Proceedings of the VLDB Endowment* 7, 8 (2014), 637–648.

[47] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. 1435–1446.

[48] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

[49] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.

[50] Uri Stemmer. 2021. Locally private k-means clustering. *Journal of Machine Learning Research* 22, 1 (2021), 7964–7993.

[51] Uri Stemmer and Haim Kaplan. 2018. Differentially private k-means with constant multiplicative error. *Advances in Neural Information Processing Systems* 31 (2018), 1–11.

[52] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing local sensitivities of counting queries with joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 479–494.

[53] Salil Vadhan. 2017. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer, 347–450.

[54] Royce J. Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2020. Differentially private SQL with bounded user contribution. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020), 230–250.

[55] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering* 23, 8 (2010), 1200–1214.
[56] Jianzhe Yu, Wei Dong, Juanru Fang, Dajun Sun, and Ke Yi. 2024. DOP-SQL: A general-purpose, high-utility, and extensible private SQL system. In *Proceedings of the International Conference on Very Large Data Bases*.
[57] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 731–745.
[58] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. 587–595.

Received 24 March 2023; revised 13 December 2023; accepted 9 September 2024